Aalto-yliopisto
Sähkötekniikan korkeakoulu
Tietoliikennetekniikan koulutusohjelma

Jouko Lehtonen

Hajautetun osoitepalvelun suunnittelu haasteelliseen verkkoon

Diplomityö
Espoo, 19. toukokuuta 2014

Valvoja: Prof. Raimo Kantola
Ohjaaja: TkL Marko Luoma

Aalto-yliopisto
Sähkötekniikan korkeakoulu
Tietoliikennetekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Jouko Lehtonen		
Työn nimi:	Hajautetun osoitepalvelun suunnittelu haasteelliseen verkkoon		
Päiväys:	19. toukokuuta 2014	**Sivumäärä:**	viii + 83
Professuuri:	Tietoliikenne- ja tietoverkkotekniikan laitos (Comnet)	**Koodi:**	S-38
Valvoja:	Prof. Raimo Kantola		
Ohjaaja:	TkL Marko Luoma		

Tässä työssä esitetään suunnitelma haasteellisessa verkossa toimivalle osoitepalvelulle.

Suurin osa nykypäivänä kehitetyistä verkkopalveluista kohdistuu luotettaviin verkkoihin. Näissä verkkoympäristöissä on tyypillisesti riittävästi siirtokapasiteettia, pienet viiveet ja vain vähän katkoja. Tällaiset ominaisuudet eivät kuitenkaan ole välttämättä voimassa haasteellisissa verkoissa. Näissä verkoissa perinteiset viestintämenetelmät toimivat heikosti tai eivät lainkaan.

Useat organisaatiot tarjoavat jäsenilleen jonkinlaisen osoitepalvelun. Palvelut pohjautuvat yleensä jollekin hakemistopalvelinjärjestelmälle ja niiden tarkoituksena on mahdollistaa osoitetietojen hakeminen ja muokkaaminen. Ne eivät kuitenkaan sellaisenaan toimi haasteellisissa verkoissa. Jotta osoitetieto olisi saatavilla haasteellisissakin ympäristöissä, on palvelu suunniteltava näiden verkkojen rajoitteet huomioon ottaen.

Suunnittelu aloitetaan tutustumalla haasteellisten verkkojen käyttötarkoituksiin ja rajoitteisiin. Tämän jälkeen tutkitaan olemassa olevia osoitepalveluteknologioita sekä -ohjelmistoja. Työssä käydään läpi myös erilaisia tietomalleja sekä replikointimenetelmiä. Kerätyn tiedon pohjalta esitetään ehdotus osoitepalvelun toteutusmallille. Malli kattaa teknologiavalinnat, viestintärajapinnat sekä tietojen replikointi- ja synkronointimenetelmät. Lopuksi työssä simuloidaan esitettyä synkronointimallia ja analysoidaan saatuja tuloksia.

Asiasanat:	haasteelliset verkot, osoitepalvelu, hakemistopalvelu, LDAP, replikointi, synkronointi
Kieli:	Suomi

Aalto University
School of Electrical Engineering
Degree Programme in Communications Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Jouko Lehtonen		
Title:	Designing a Distributed Address Service for a Challenged Network		
Date:	May 19, 2014	**Pages:**	viii + 83
Professorship:	Department of Communications and Networking (Comnet)	**Code:**	S-38
Supervisor:	Prof. Raimo Kantola		
Instructor:	Lic.Sc. (Tech.) Marko Luoma		

In this thesis, we present a design for an address service that can be used in a challenged network.

Most network services developed today are directed at reliable networks. Typically, these network environments provide enough transfer capacity, low latencies and only few disconnections. These properties, however, are not in effect in challenged networks. In these networks, traditional means of communication work poorly or not at all.

Many organizations provide an address service to their members. The services are usually based on a directory service system and their purpose is to enable the searching and modifying of address data. They do not, however, work in challenged networks as is. In order to make address data available in challenged environments the limitations of the network need to be taken into consideration while designing the service.

The designing begins with an introduction to the purposes and limitations of challenged networks. Next, we examine some of the existing address service technologies and software implementations. The thesis also covers different kinds of data models and replication methods. Based on the gathered information, we propose an implementation model for the address service. The model covers technology choices, communication interfaces as well as replication and synchronization methods. Finally, the proposed synchronization model is simulated and the results are analyzed.

Keywords:	challenged networks, address service, directory service, LDAP, replication, synchronization
Language:	Finnish

Sisältö

Lyhenteet

ACAP	Application Configuration Access Protocol
ACL	Access Control List
AD LDS	Active Directory Lightweight Directory Services
ADAM	Active Directory Application Mode
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BDB	Berkeley Database
DAP	Directory Access Protocol
DAV	Distributed Authoring and Versioning
DISP	Directory Information Shadowing Protocol
DIT	Directory Information Tree
DN	Distinguished Name
DOP	Directory Operational Binding Management Protocol
DSA	Directory System Agent
DSML	Directory Services Markup Language
DSP	Directory System Protocol
DTN	Delay-Tolerant Networking
DUA	Directory User Agent
EBNF	Extended Backus-Naur Form
GAL	Global Address List
HAB	Hierarchical Address Book
HDB	Hierarchical Database
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
IMSP	Internet Messaging Support Protocol
IP	Internet Protocol
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
MDB	Memory-Mapped Database
ODSEE	Oracle Directory Server Enterprise Edition

OID	Object Identifier tai Oracle Internet Directory
OSI	Open Systems Interconnection
OUD	Oracle Unified Directory
RDN	Relative Distinguished Name
Regex	Regular expression
RFC	Request for Comments
Slapd	Stand-alone LDAP Daemon
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAN	Wide Area Network
WebDAV	Web-based Distributed Authoring and Versioning
XML	Extensible Markup Language

Luku 1

Johdanto

Tässä työssä käsitellään hajautetun osoitepalvelun suunnittelemista haasteelliseen verkkoon. Nykypäivänä suurin osa verkkopalveluista tuotetaan luotettaviin verkkoihin. Luotettavilla verkoilla tarkoitetaan ympäristöjä, joissa esiintyy vain vähän katkoksia, viiveet ovat pieniä ja siirtokapasiteettia sekä laitteiden laskentatehoa on riittävästi. Valtaosa luotettavissa verkoissa toimivista palveluista käyttää viestintään IP-protokollaa (Internet Protocol) ja TCP-yhteyksiä (Transmission Control Protocol). Näiden protokollien toiminta perustuu erilaisille olettamuksille osoitteistuksesta, siirtoviiveistä sekä yhteydellisyydestä.

On kuitenkin olemassa ympäristöjä, joissa luotettavien verkkojen lainalaisuudet eivät päde. Tällaisia verkkoja sanotaan haasteellisiksi verkoiksi. Niissä ei ole mahdollista käyttää samoja protokollia ja palveluita, jotka toimivat ongelmitta luotettavissa verkoissa. Esimerkkejä haasteellisista ympäristöistä ovat muun muassa sotilasverkot ja suuriviiveinen avaruusviestintä.

Nykyään varsinkin monissa suurissa yrityksissä ja organisaatioissa on käytössä jonkinlainen osoite- ja organisaatiopalvelu. Palvelun tehtävä on tarjota sen käyttäjille yhteystietoja kuten puhelinnumeroita ja sähköpostiosoitteita. Lisäksi palvelu voi sisältää tietoja henkilöstöhierarkiasta. Tyypillisesti tällaiset palvelut on toteutettu LDAP-palvelimilla (Lightweight Directory Access Protocol). Suurimmissa organisaatioissa tiedot on hajautettu useammalle palvelimelle.

Haasteellisissa ympäristöissä LDAP-viestintä ei kuitenkaan yleensä onnistu, sillä käytössä ei ole vaadittua TCP-yhteyttä. Tästä syystä osoitekyselyitä ja -päivityksiä varten on käytettävä muita protokollia. Lisäksi osoitepalvelun verkkorajapinta on suunniteltava ottaen huomioon haasteellisten verkkojen asettamat rajoitukset esimerkiksi siirtokapasiteetille.

Jotta osoitetiedot pysyisivät ajan tasalla, täytyy osoitepalvelun mahdollistaa hakujen lisäksi myös tietojen päivitys. Asiakkaiden täytyy siis pystyä muokkaamaan palvelun tietoja. Hajautetussa järjestelmässä ei kuitenkaan riitä, että tiedot päivitetään yhdellä laitteella, vaan muutokset on replikoitava muualle verkkoon. Monet hakemistopalvelinohjelmistot tarjoavatkin mahdollisuuden tietojen replikointiin, mutta järjestelmät vaativat toimiakseen luotettavan verkon.

Luotettavissa verkoissa käytössä olevat replikointimenetelmät soveltuvat haasteellisiin ympäristöihin huonosti. Tyypilliset replikointimekanismit toimivat riittävän hyvin pieniviiveisissä verkoissa, mutta haasteellisten verkkojen viiveet ja linkkien epäluotettavuus voivat aiheuttaa ongelmia eri osoitepalvelinten tietojen yhdenmukaisuudessa ja eheydessä. Replikointijärjestelmään on siis toteutettava mekanismeja, joilla ongelmatilanteita voidaan välttää tai ratkaista.

1.1 Työn tavoitteet

Tässä työssä perehdytään haasteellisten verkkojen rajoitteisiin sekä tutustutaan olemassa oleviin osoitepalveluohjelmistoihin ja -protokolliin. Työssä arvioidaan myös erilaisten tietomallien soveltuvuutta osoitepalveluun. Lisäksi tarkoituksena on esitellä erilaisia replikointimenetelmiä ja arvioida niiden soveltuvuutta haasteelliseen verkkoon sekä käydä läpi joitakin tällaisissa verkoissa toimivia replikointijärjestelmiä.

Työn perimmäisenä tavoitteena on suunnitella näiden tietojen pohjalta hajautettu osoitepalvelujärjestelmä, joka toimii haasteellisessa verkossa. Työssä valitaan hierarkkiselle organisaatiotiedolle tietomalli ja pääsynhallintalogiikka, joka toimii myös hajautetussa ympäristössä. Osoitepalvelun asiakasrajapinnan on mahdollistettava osoitetietojen haku- ja päivitystoiminnot. Lisäksi tarkoituksena on suunnitella replikointijärjestelmä, joka toimii hajautetussa haasteellisessa verkossa mahdollisimman sujuvasti pyrkien eri tietokantapalvelinten yhdenmukaisuuteen. Yhdenmukaisuuden varmistamiseksi palveluun suunnitellaan myös synkronointimekanismi, jolla erilaiset hakemistot saatetaan samaan tilaan pienellä viesti- ja liikennemäärällä.

1.2 Työn rakenne

Luvussa 2 tarkastellaan haasteellisia verkkoja, niiden yleisimpiä käyttöympäristöjä sekä ominaispiirteitä. Luku 3 käsittelee olemassa olevia ratkaisuja osoitepalvelujen toteuttamiseen. Ratkaisut kattavat sekä osoitetiedon jakeluun suunnitellut

protokollat että kyseisiä protokollia käyttävät sovellukset. Luvussa 4 käydään läpi osoitepalvelun kannalta olennaisia tietomalleja ja datan replikointiin keskittyviä tietoliikennevalintoja. Ohessa esitellään joitakin haasteellisessa verkossa toimivia replikointiratkaisuja. Luvussa 5 esitetään ehdotus osoitepalvelun toteutusmallista. Malli kattaa palvelun tietomallin sekä verkkoviestinnän, johon kuuluvat asiakas-, replikointi- ja synkronointirajapinnat. Verkkoon suunniteltua synkronointimallia testataan simulaatioympäristössä luvussa 6. Lopuksi luvussa 7 tehdään vielä yhteenveto työstä.

Luku 2

Haasteelliset verkot

Haastellisilla verkoilla tarkoitetaan tietoliikenneverkkoja, joissa tiedonsiirtoa hankaloittavat erilaiset ympäristötekijät. Tällaisiin verkkoihin turvaudutaan silloin, kun luotettavaa tiedonsiirtomenetelmää ei ole tarjolla. Tässä luvussa esitellaan ensin joitakin haasteellisten verkkojen käyttöympäristöjä, minkä jälkeen käydään vielä läpi näiden verkkojen ominaispiirteitä.

2.1 Käyttöympäristöt

Nykyiset tietokoneverkot käyttävät useimmiten viestintään IP-protokollaa sekä TCP:tä tai UDP:tä (User Datagram Protocol). Hyvin standardoitu ja laajalti käytetty IP mahdollistaa useiden erilaisten teknologioiden käytön alemmilla verkkotasoilla. IP-yhteyksien sujuva toiminta kuitenkin asettaa monenlaisia vaatimuksia näille verkkotasoille. Vaatimuksiin lukeutuvat muun muassa yhteys päästä päähän, riittävän pieni viestien vastausaika ja tarpeeksi pieni pakettihävikki. Internetissä ja muissa tietokoneverkoissa nämä kriteerit useimmiten täyttyvät, mutta haasteellisissa verkoissa samat olosuhteet eivät välttämättä päde. Alla on käyty läpi erilaisia haasteellisia verkkoja ja niiden rajoitteita sekä erityisvaatimuksia. [12]

Maanpäälliset mobiiliverkot

Radioverkkojen ominaispiirteet tekevät joistakin langattomista verkoista haasteellisen ympäristön. Mobiiliverkoissa tiedonsiirtoa vaikeuttavat useat eri tekijät. Verkon kapasiteettia rajoittaa huomattavasti radiointerferenssi, jota syntyy muun muassa muista alueella toimivista mobiililaitteista. Tämän lisäksi yhteyden toimi-

vuutta haittaavat radioaaltojen heikentyminen rakennusten ja maastonmuotojen takia sekä signaalin heijastuminen eri pinnoista. [35]

Jotkin mobiiliverkot saattavat altistua merkittäville signaalin voimakkuuden muutoksille sekä verkkosolmujen liikkeelle. Seurauksena verkko voi jakautua erillisiin saarekkeisiin. Jakautuminen voi olla ennakoimatonta, jos verkkolaitteet liikkuvat sattumanvaraisesti. Joissakin tapauksissa verkon jakautumista voidaan taas ennakoida. Tämä pätee esimerkiksi tilanteessa, jossa linja-autoa käytetään viestejä välittävänä kytkimenä. Tällainen järjestely mahdollistaa viestien siirtämisen pitkiäkin matkoja käyttäen ainoastaan lyhyen kantaman yhteyksiä. [12]

Epätavallisten siirtoteiden verkot

Joissakin erikoistapauksissa ei ole mahdollista käyttää perinteisiä siirtoteitä. Näitä tapauksia ovat esimerkiksi vedenalaiset ja erittäin pitkän kantaman ympäristöt. Tällöin voidaan viestiä esimerkiksi akustisten linkkien tai satelliittien välityksellä. Näissä ympäristöissä siirtoviiveet ja katkokset voivat olla pitkiä ja arvaamattomia. Järjestelmien toimintaan vaikuttavat muun muassa sääolosuhteet sekä taivaankappaleiden liikkeet. [12]

Sotilasverkot

Sotilaskäyttöön tarkoitetut verkot altistuvat useille ongelmille, jotka tekevät viestinnästä haasteellista. Tiedonsiirtoon vaikuttavat verkkolaitteiden liikkuminen, erilaiset ympäristötekijät sekä vihollisjoukkojen suorittama viestiliikenteen häirintä. Näiden lisäksi sotilasverkoissa eri palveluiden liikennettä siirretään usein eri prioriteeteillä, jolloin kiirettömämpi liikenne saa vähemmän resursseja. Tämä voi pienentää verkossa toimivan palvelun jo ennalta rajallista kaistanleveyttä ja täten kasvattaa siirtoviiveitä. [12]

Sensoriverkot

Erilaiset sensoriverkot toimivat myös usein haasteellisissa ympäristöissä. Päätelaitteilla on tyypillisesti hyvin rajalliset resurssit esimerkiksi laskentatehon, muistin ja virran suhteen. Lisäksi sensoriverkot ovat usein varsin suuria ja voivat koostua jopa miljoonista verkkolaitteista. [12]

2.2 Ominaispiirteet

Haasteelliset verkot toimivat ympäristöissä, jotka poikkeavat merkittävästi perinteisistä verkkoympäristöistä. Erityishaasteita ja -vaatimuksia verkkoprotokollille asettavat muun muassa viiveet, alhainen siirtokapasiteetti sekä päätelaitteiden erilaiset ominaispiirteet. Haasteellisissa verkoissa eivät siis päde samat lainalaisuudet kuten esimerkiksi Internetissä tai kännykkäverkoissa. Tässä osiossa käydään läpi erilaisia ongelmia ja rajoituksia, jotka on otettava huomioon haasteellisen verkon viestintää suunniteltaessa. [12]

Korkea viive, matala siirtokapasiteetti

Monissa haasteellisissa verkoissa siirtoviiveet voivat olla suuria sekä tiedonsiirtonopeudet alhaisia. Käytettävissä oleva kaista saattaa olla esimerkiksi vain joitakin kilobittejä sekunnissa ja verkkoviive joitakin sekunteja. Lisäksi linkkien siirtonopeudet eri suuntiin voivat erota huomattavasti. Äärimmillään siirtotie toiseen suuntaan voi puuttua kokonaan. [12]

Yhteyden epäluotettavuus

Yhteyden katkeaminen on monissa haasteellisissa verkoissa hyvin yleistä. Katkokset voidaan luokitella karkeasti kahteen ryhmään: vioista johtuviin ja muihin katkoihin. Vikapohjaiset katkokset ovat tuttuja jo luotettavista verkoista. Muut katkokset puolestaan johtuvat tietyistä haasteellisten verkkojen ominaispiirteistä kuten langattomien laitteiden siirtymisestä ja alhaisista toimintasykliajoista. [12]

Pitkät jonotusajat

Perinteisissä pakettiverkoissa usean hypyn yli siirrettävien pakettien viiveet muodostuvat suurelta osin jonotusajoista. Jonotusajat ovat tyypillisesti hyvin pieniä, ja paketit hylätään, jos seuraava reititin ei ole heti saavutettavissa. Haasteellisissa verkoissa puolestaan jonotusajat voivat olla todella suuria. Jos verkkolaitteiden väliset yhteydet katkeavat pitkiksi ajoiksi, saattavat jonotusajat yltää tunteihin tai jopa päiviin. Koska uudelleenlähetys haasteellisissa verkoissa voi olla ongelmallista, on usein helpompaa säilyttää viestejä reitittimien jonoissa. Tämä kasvattaa viestijonoja entuudestaan. [12]

Yhteentoimivuusongelmat

Yhteentoimivuus useimmissa haasteellisissa verkoissa on melko alhainen, ja verkkoarkkitehtuurit niissä koostuvatkin lähinnä siirtotien pääsynhallintaprotokollista. Monissa haasteellisissa verkoissa pelkkä viestinnän mahdollistaminen on edelleen kehityksen alla, joten laajempien protokollapinojen käyttöönotto ei ole lainkaan toteutettavissa. Vähäisten resurssien laitteet joutuvat usein nojautumaan protokollatoteutuksissa erityyppisiin oikoteihin, mikä hankaloittaa yhteensopivuutta erilaisten verkkoteknologioiden kanssa. [12]

Tietoturva

Koska haasteellisissa verkoissa tietyt verkkoresurssit ovat hyvin rajoitettuja, tulisi niitä suojata väärinkäytöksiltä. Käytännössä ainakin käyttäjien autentikointi sekä pääsynhallinta siirtoteille olisi syytä suojata verkon kriittisissä pisteissä. Myös eri palvelutasot on suojattava, mikäli verkko sellaisia tukee. Päästä päähän -pohjaiset ratkaisut vaativat usein jonkinlaisen avaimenvaihtomekanismin, joten ne eivät tyypillisesti sovi suuriviiveisiin ja yhteysongelmaisiin verkkoihin. Autentikointi olisi suotavaa suorittaa jo päätelaitteiden liitospisteissä, jotta päästä päähän -liikenne voidaan pitää alhaisena. [12]

Rajoitettu toimintaikä

Joissakin tapauksissa päätelaitteet joutuvat toimivaan vihamielisissä ympäristöissä, minkä seurauksena niiden toimintaiät voivat olla varsin lyhyitä. Tämä pätee erityisesti sensori-, sotilas- ja hälytyshenkilöstöverkkoihin. Viestien kuljetusaika tällaisissa verkoissa voi olla pidempi kuin päätelaitteen toimintaikä. Viestien kuittausta ei siis voida tehdä perinteisellä päästä päähän -mallilla, koska vastaanottaja saattaa poistua verkosta ennen viestin saapumista. Näissä tapauksissa viestinvälityksen varmistus on määrättävä muiden, toiminnassa pysyvien verkkolaitteiden tehtäväksi. [12]

Matala toimintasykliaika

Jotkin laitteet kuten erilaiset sensorit saavat virtansa akuista, minkä takia niiden toimintasyklit ovat poikkeuksellisen lyhyet. Tästä johtuen niiden verkkoviestinnän toimintamalli määritelläänkin yleensä etukäteen eivätkä ne aina reagoi verkkotapahtumiin välittömästi. Tällainen toiminta puolestaan asettaa tietynlaisia haasteita muun muassa reititykselle. [12]

Rajoitetut resurssit

Monet haasteellisten verkkojen päätelaitteet saattavat joutua toimimaan rajoitetuilla resursseilla kuten vähäisellä muistin määrällä tai laskentateholla. Tällöin esimerkiksi viestien säilyttäminen pitkiä aikoja päätelaitteen muisteissa voi hankaloittaa laitteen muita toimintoja. [12]

Luku 3

Osoitepalvelujärjestelmät

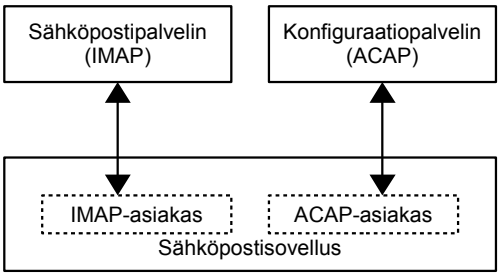
Osoitepalvelu voi tuoda merkittävästi lisäarvoa viestintäsovelluksille. Tästä syystä on olemassa useita teknologioita, joilla kyseinen palvelu voidaan toteuttaa. Tämän luvun tarkoitus on esitellä erilaisia osoitepalveluissa käytettyjä protokollia sekä yleisimpiä palvelinohjelmistoja.

3.1 Protokollat

Tässä osiossa esitellään joitakin osoitepalveluiden pohjana käytettyjä verkkoprotokollia. Tarkasteltavana ovat sähköpostin rinnalle kehitetty ACAP (Application Configuration Access Protocol), HTTP:lle (Hypertext Transfer Protocol) pohjautuva CardDAV sekä hakemistostandardit X.500 ja LDAP (Lightweight Directory Access Protocol).

3.1.1 ACAP

ACAP [22] on TCP:llä toimiva sovellustason protokolla, joka suunniteltiin alunperin IMAPin (Internet Message Access Protocol) rinnalle sovellusasetusten etätallentamiseen ja -lukemiseen. Käytännössä tämä tarkoittaa esimerkiksi käyttäjien sähköpostiasetusten säilyttämistä palvelimella, josta tiedot voidaan helposti hakea paikasta riippumatta. Protokolla ei kuitenkaan rajoitu ainoastaan konfiguraatiotietoon, vaan myös esimerkiksi osoitekirjojen tallentaminen on mahdollista. ACAP suunniteltiin korvaamaan IMSP [21] (Internet Messaging Support Protocol), joka ei edennyt standardiksi asti. ACAPin ja IMAPin yhteistoiminta sähköpostisovelluksen kanssa on hahmoteltu kuvassa 3.1. [41]

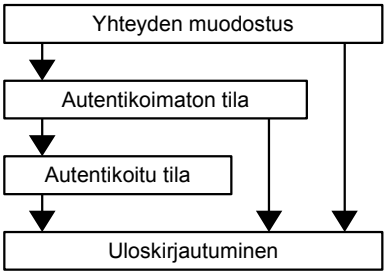


Kuva 3.1: ACAP- ja IMAP-palvelinten käyttö sähköpostisovelluksen kanssa. [41]

ACAP on monella tavalla LDAPin kaltainen protokolla. Kummatkin perustuvat asiakas-palvelin -malliin, toimivat hierarkkisen hakemiston kanssa, käyttävät avain-arvo -pareja ja tarjoavat monipuolisen hakurajapinnan. Asiakasohjelmissa ACAP on yksinkertaisempi toteutettava kuin palvelimen puolella. Tämän suunnitteluratkaisun tarkoitus on kannustaa ACAP-asiakasohjemien luomiseen. [41]

Tietoliikenne

ACAP on asiakas-palvelin -protokolla. Se toimii TCP:llä käyttäen oletuksena porttia 674. Kukin istunto voi olla yhdessä kolmesta tilasta. Nämä ovat autentikoimaton tila, autentikoitu tila ja uloskirjautumistila. Asiakkaan muodostaessa yhteyttä palvelin menee autentikoimattomaan tilaan. Käyttäjän tunnistautuessa palvelin siirtyy autentikoituun tilaan, jolloin useimmat asiakkaan komennot muuttuvat sallituiksi. Tällöin asiakassovellus voi suorittaa hakuja ja tehdä muutoksia hakemistoon. Lopuksi yhteyttä suljettaessa istunto käy vielä uloskirjautumistilassa. Tilakaavio on esitetty kuvassa 3.2. [41]



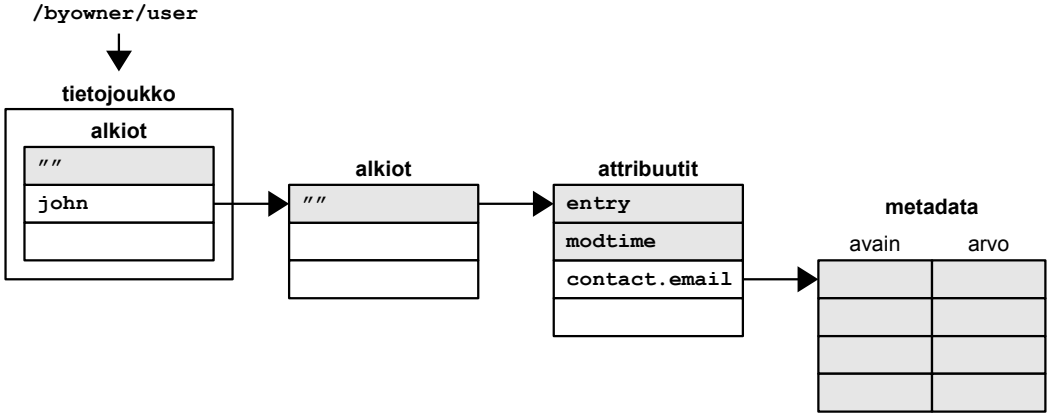
Kuva 3.2: ACAP-istunnon tilakaavio. [22]

ACAP-asiakassovellusten täytyy spesifikaation mukaan toimia myös ilman verkkoyhteyttä. Tällöin käsiteltävät tiedot säilytetään asiakkaan välimuistissa ja muutokset päivitetään palvelimelle seuraavalla yhteyskerralla. Verkon toimiessa asiakassovellukset tyypillisesti ottavat yhteyden palvelimelle käynnistyessään ja tarkistavat paikallisen tiedon oikeellisuuden. Myöhemmin asiakasohjelmat muodostavat palvelinyhteyksiä päivittääkseen muuttuneita tietoja. [41]

Tietorakenteet

ACAP-palvelimen tiedot säilytetään hierarkkisessa varastossa, jossa tasot erotellaan kauttaviivalla. Varasto koostuu tietojoukoista, alkioista, attribuuteista ja metadatasta. [41]

Tietojoukko vastaa yhtä hierarkian tasoa ja pitää sisällään alkioita. Käytännössä tietojoukko on vain alkion erikoistapaus, johon voidaan säilöä muita alkioita. Tietojoukko muodostetaan yksinkertaisesti kirjoittamalla alkion subdataset-attribuuttiin pistemerkki. Alkiot ovat hakemiston yksittäisiä merkintöjä, jotka sisältävät attribuutteja. Kullakin alkiolla on kolme ennalta määriteltyä attribuuttia, mutta lisää voi tarvittaessa luoda. Ennalta määritellyt attribuutit ovat alkion nimi, muokkausaika sekä tietojoukkotoiminnallisuuden ilmoittava totuusarvo. Attribuutit puolestaan ovat alkioiden ominaisuuskuvauksia. Ne koostuvat metadatasta, mikä käytännössä tarkoittaa joukkoa nimi-arvo -pareja. Metadatassa säilytetään attribuutin nimi, arvot, koko, pääsynhallintalista sekä kirjautuneen asiakkaan oikeudet kyseiseen attribuuttiin. ACAPin tietojoukkojen rakenne näkyy kuvassa 3.3. [41]



Kuva 3.3: ACAPin tietorakenne. [41]

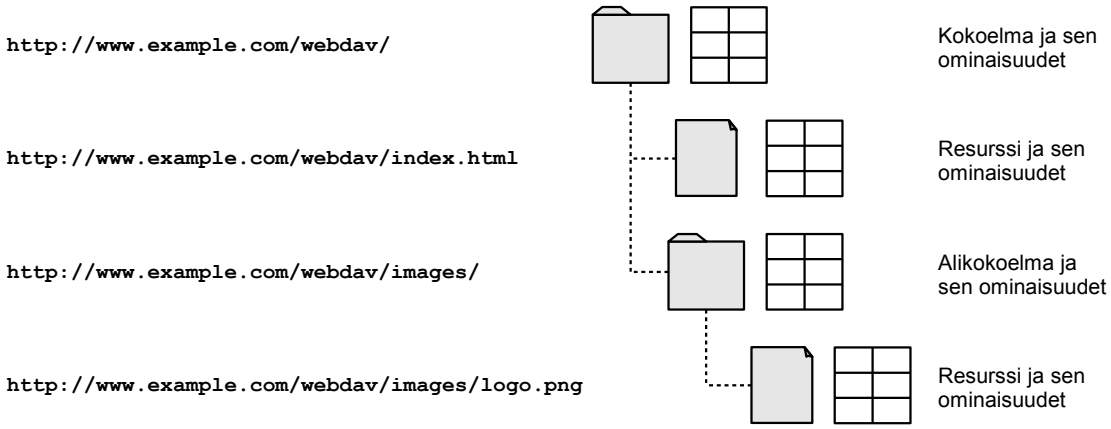
ACAP mahdollistaa hyvinkin tarkan pääsynhallinnan määrittelemisen. Pääsynhallintalistat säilytetään attribuuteissa. Tunnuksille voidaan antaa oikeudet hakemiseen, lukemiseen, kirjoittamiseen, muokkaamiseen ja pääsynhallinnan muuttamiseen. Oikeuksia voidaan jakaa attribuuttikohtaisesti tai laajemmin käyttäen perintää. [41]

3.1.2 WebDAV ja CardDAV

WebDAV

WebDAV [11] (Web-based Distributed Authoring and Versioning) on web-pohjainen sisällönhallintaprotokolla. Se laajentaa HTTP-protokollaa erilaisilla toiminnoilla, joiden avulla web-palvelimen sisältöä voidaan muokata etäältä. WebDAV siis mahdollistaa palvelimen käyttämisen etätiedostojärjestelmänä. [10]

WebDAV-palvelimen sisältö on tavallisen web-palvelimen kaltainen. Tietorakenne on hierarkkinen ja koostuu kokoelmista sekä resursseista, joihin osoitetaan URL:illä (Uniform Resource Locator). Käytännössä kokoelmalla tarkoitetaan hakemistoa ja resurssilla joko tiedostoa tai hakemistoa. Esimerkki tietorakenteesta näkyy kuvassa 3.4. Lisäksi resursseilla on ominaisuuksia, jotka ovat nimi-arvo -pareja. Ominaisuudet voivat ilmaista esimerkiksi resurssin kielen tai luontipäivämäärän. Ominaisuuksien tallennuspaikka riippuu palvelintoteutuksesta. Tallennuspaikkana voi toimia vaikka tietokanta tai itse tiedostojärjestelmä. Resursseja voidaan myös lukita, jotta useampi käyttäjä ei muokkaisi niitä samanaikaisesti. [10]



Kuva 3.4: WebDAV-palvelimen tietorakenne. [10]

Koska HTTP:n määrittelemät metodit eivät riitä monipuoliseen tiedostojen hallintaan, on WebDAViin lisätty joitakin toimintoja. Nämä toiminnot mahdollistavat resurssien kopioinnin ja siirtämisen sekä kokoelmien luomisen. Resurssien ominaisuuksien hallintaan määritellään myös uusia metodeja. Niiden avulla ominaisuuksia voidaan muokata ja hakea. Tällöin viestintä tapahtuu XML-muodossa (Extensible Markup Language). Lisäksi WebDAVissa on operaatioita tiedostolukkojen hallintaan. [10]

RFC 3744 määrittelee WebDAVille tuen pääsynhallintalistoille. Pääsynhallintasäännöt tallennetaan palvelimelle kunkin resurssin ominaisuuksiin. Säännöt määrittelevät käyttäjät ja ryhmät, joita rajoitukset koskevat, sekä oikeuksien tason. [4]

CardDAV

CardDAV on WebDAViin pohjautuva osoitekirjaprotokolla, joka määrittelee tavan hallita ja jakaa osoitetietoja. Se rakentuu samoille käsitteille kuin WebDAV. [7]

CardDAV pohjautuu WebDAVin tavoin HTTP:hen ja käyttää sisällön muokkaamiseen olemassa olevia HTTP- sekä WebDAV-metodeja. Myös CardDAV hyödyntää viestinnässään XML-formaattia. Koska CardDAV pohjautuu WebDAVin tavoin HTTP-protokollaan, on useissa laitteissa jo olemassa sovellus, jolla osoitetietoja voidaan selata. Uudet metodit on kuitenkin toteutettava erikseen. [7]

CardDAV-palvelimen tietorakenne perustuu samanlaiselle hierarkialle kuin WebDAV. Itse osoitetiedot välitetään vCard-formaatissa [33]. Standardinmukaisten vCard-attribuuttien lisäksi on mahdollista käyttää epästandardeja ominaisuuksia. Omia attribuutteja voidaan luoda nimeämällä avaimet käyttäen X-alkuliitettä. [7]

CardDAViin on suunniteltu laajennusta, joka mahdollistaisi protokollan käyttämisen yhdyskäytävänä hakemistoon. Sen avulla CardDAV-palvelimen kautta voitaisiin hakea esimerkiksi osoitetietoja LDAP-hakemistosta. Yhdyskäytävä ei kuitenkaan sallisi tietojen muokkaamista, ainoastaan lukemisen. [6]

3.1.3 X.500

X.500 on standardijoukko, joka määrittelee hakemistopalvelun. X.500-pohjaisia järjestelmiä on käytetty muun muassa osoitekirjojen toteuttamiseen yritysympäristöissä. X.500:n ongelmaksi kuitenkin muodostuivat vaatimus OSI-mallin (Open Systems Interconnection) mukaisesta protokollapinosta sekä järjestelmän monimutkaisuus. Monet X.500-hakemistoja käyttäneet tahot olivat siirtyneet yleistyneisiin IP-järjestelmiin ja kaipasivat kevyempää hakemistoprotokollaa. Tätä varten kehitettiin LDAP. [9]

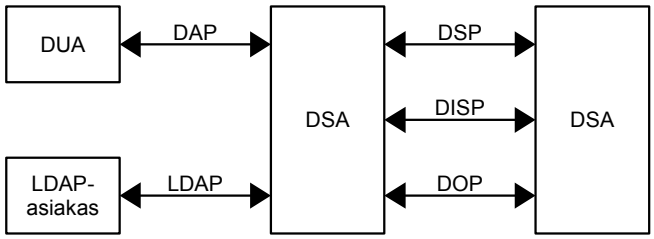
X.500-hakemiston tietomalli on pääosin sama kuin LDAP-palvelinten käyttämien hakemistojen. Tieto järjestetään olioista koostuvaan hierarkkiseen puuhun. Olioilla on yksilöllinen DN-nimi (Distinguished Name) sekä attribuutteja. Attribuuteilla puolestaan on tietotyyppejä, joille voidaan määritellä vertailusääntöjä. [6]

Tietoliikenne

X.500-järjestelmän asiakassovelluksista käytetään nimitystä DUA (Directory User Agent) ja palvelimista DSA (Directory System Agent). DUA:t viestivät palvelimille käyttäen DAPia (Directory Access Protocol), kun taas palvelimet käyttävät keskinäiseen viestintäänsä DSP:tä (Directory System Protocol). [6]

DAP koostuu operaatioista, joilla hakemiston tietoja voidaan lukea ja kirjoittaa. Operaatioiden toiminnallisuus on hyvin samanlainen kuin LDAP-protokollassa. DSP ja DAP ovat myös hyvin samankaltaisia. DSP:n tarkoitus on mahdollistaa DSA:iden välinen koordinointi, jotta DUA:n lähettämään kyselyyn voidaan palauttaa vastaus tapauksessa, jossa ensimmäinen palvelin ei pysty vastausta antamaan. Palvelimet siis ketjuttavat operaatioita toisilleen tarvittaessa. [6]

DSP:n ja DAPin lisäksi X.500 käyttää kahta muuta verkkoprotokollaa: DOP (Directory Operational Binding Management Protocol) ja DISP (Directory Information Shadowing Protocol). DOPia käytetään palvelinten välillä konfiguraatiotiedon välittämiseen. Sen avulla voidaan välittää hakemistotietojen hallintaan ja replikointiin liittyvää tietoa. DISP:tä käytetään hakemistotietojen replikointiin palvelinten välillä. Kuvassa 3.5 on esitetty X.500-järjestelmän käyttämät viestintäprotokollat. [6]



Kuva 3.5: X.500-järjestelmän tietoliikenneprotokollat. [26]

---

[6] Chadwick, D. W. Understanding X.500 - The Directory. `http://sec.cs.kent.ac.uk/x500book/`. Viitattu 19.4.2013.

Vaikka LDAP-pohjaiset ratkaisut ovatkin suurelta osin korvanneet tarpeen X.500:n käytölle, on useita eri X.500-palvelinohjelmistoja yhä saatavilla. [27]

3.1.4 LDAP

LDAP on sovellustason protokolla, jota käytetään hierarkkisesti järjestellyn hakemistotiedon käsittelyyn. Se pohjautuu raskaampaan X.500-standardijoukkoon, joka käsittelee hakemistopalveluita. LDAP toimii TCP/IP-protokollapinolla ja on käytössä useissa eri hakemistototeutuksissa. [19]

LDAPia ei ole suunniteltu vain yhtä tiettyä tarkoitusta varten, vaan sitä voidaan soveltaa useisiin eri ympäristöihin ja käyttötarkoituksiin. Skaalautuvuutensa vuoksi sitä käytetään sekä laajoissa julkisissa sovelluksissa että pienemmissä tietoverkoissa. Erilaisissa yrityksissä ja laitoksissa LDAP-palvelimet tarjoavat usein tietoja organisaatiosta ja sen henkilökunnasta. Hakemistossa voidaan kuitenkin säilyttää myös esimerkiksi dataa tietokoneista, verkkolaitteista ja sovelluksista. LDAP-järjestelmiä voidaan integroida useisiin eri palveluihin. Näihin lukeutuvat muun muassa sähköpostisovellukset sekä autentikointi- ja resurssienhallintajärjestelmät. [19]

LDAP-palvelimia on jo pitkään käytetty osoitekirjatietojen jakamiseen. Osoitekirjat ovatkin LDAP-sovellusten yleisin käyttötarkoitus. Useimmat sähköpostiohjelmat tukevatkin LDAP-yhteyttä osoitteiden hakemiseen, vaikka joissakin sovelluksissa on käytössä myös suljettuja standardeja. [9]

LDAP-hakemistot soveltuvat tietorakenteensa puolesta hyvin organisaatioiden hierarkian kuvaamiseen ja yhteystietojen säilyttämiseen. Hakemistopuu voidaan rakentaa organisaation osastojen ja henkilöstön mukaisesti. [9] [3] [2] [23]

Tietoliikenne

LDAP on asiakas-palvelin -protokolla, joka toimii TCP-yhteyden yli ja mahdollistaa pääsyn hakemistotietoihin. LDAP-palvelin kuuntelee asiakkaiden pyyntöjä oletuksena portissa 389 [3]. Protokollan tarjoamiin operaatioihin kuuluvat muun muassa haut ja tietojen lukeminen. Lisäksi hakemistossa olevaa dataa voidaan muokata lisäämällä, päivittämällä ja poistamalla tietoja. [9]

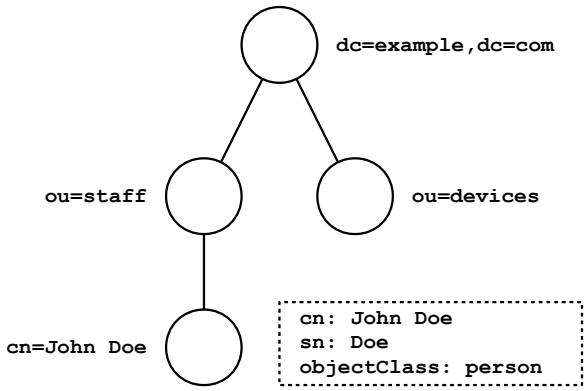
Replikointia ja synkronointia varten LDAP-protokollaa on laajennettu synkronointioperaatiolla. Operaatiolla pyritään hakemistokopioiden yhdenmukaisuuteen. Se pyrkii välttämään tarpeetonta datansiirtoa, joten ainoastaan uudet ja muuttuneet

[27] X500Standard.com. X.500 Products by Vendor. `http://www.x500standard.com/index.php?n=Participate.Vendors`. Viitattu 19.4.2013.

tiedot välitetään kopioille. Synkronointia voidaan suorittaa kahdella eri tavalla. Ensimmäisessä tapauksessa asiakassovellus voi pyytää päivitettyjä tietoja pääpalvelimelta normaalin LDAP-haun tapaan. Vanhan tiedon uudelleensiirtämisen välttämiseksi kyselyssä käytetään apuna evästettä, joka kertoo asiakassovelluksen tilan. Toisessa tapauksessa asiakas tekee pysyvän haun, jolloin palvelinohjelmiston tehtäväksi jää huolehtia muutosten välittämisestä sellaisten ilmetessä. [42]

Tietomalli

LDAP-palvelinten hakemistoissa käytetään hierarkkista tietomallia. Hakemistopuu (DIT, Directory Information Tree) muodostuu hierarkkisesti järjestetyistä solmuista eli alkioista. Kuvassa 3.6 näkyy LDAP-tietomallin mukainen puurakenne. [3]



Kuva 3.6: Esimerkki yksinkertaisesta LDAP-hakemistopuusta. [3]

Tietoalkiot ovat olioita, jotka pohjautuvat hakemiston skeemassa määriteltyihin luokkiin. Yksi alkio voi johtua yhdestä tai useammasta olioluokasta. Luokat määrittelevät, mitä attribuutteja oliolla voi ja täytyy olla. [3]

Attribuuttien tarkoitus on säilyttää arvoja. LDAPissa attribuutit voivat olla yksi- tai moniarvoisia. Moniarvoiset attribuutit ovat hyödyllisiä esimerkiksi tapauksissa, joissa yhdelle alkiolle halutaan tallentaa vaikkapa useampi puhelinnumero [3]. Moniarvoisten attribuuttien arvot säilytetään ja palautetaan kuitenkin järjestämättöminä, joten esimerkiksi ensisijaista puhelinnumeroa ei voida saadusta listasta varmuudella päätellä. [9]

Kullakin hakemiston alkiolla on oma RDN-nimi (Relative Distinguished Name), joka koostuu yhdestä tai useammasta olion attribuutista. RDN-nimeä käytetään

alkion tunnisteena, jonka avulla sisarsolmut erotetaan toisistaan. Pelkkä RDN ei kuitenkaan riitä olion yksilöimiseen hakemistossa, vaan tätä varten käytetään DN-nimeä (Distinguished Name). DN koostuu useista RDN-nimistä ja toimii polkuna, joka osoittaa olion paikan hierarkiassa. [3]

Skeema määrittelee, mitä tietoja hakemistoon voidaan tallentaa. Se koostuu olioluokka- ja attribuuttimääritelmistä. LDAP-palvelimia varten on olemassa joukko standardiskeemoja, joissa määritellään yleisimpiä tietorakenteita kuten henkilö ja organisaatio. Standardeja oliokuvauksia käyttämällä on helpompi saada eri ohjelmistot toimimaan yhdessä. On kuitenkin mahdollista luoda uusia olioita ja attribuutteja tarpeen mukaan. [9]

Attribuuttimääritelmät koostuvat nimestä, oliotunnuksesta (OID, Object Identifier), syntaksista, vertailusäännöistä ja periyttämisestä. Nimi kuvaa tallennettavaa tietoa, ja oliotunnus on uniikki numerosarja. Syntaksi kertoo attribuutin tietotyypin, ja vertailusäännöt määrittävät, miten haut suoritetaan. Periyttämistä käyttämällä samankaltaisten attribuuttien ominaisuuksia ei tarvitse määritellä aina erikseen. [9]

Olioluokat määrittävät, minkälaisia alkioita hakemistoon voidaan tallentaa. Kullakin luokalla on nimi, tunnus (OID) ja periyttämistieto. Nämä ominaisuudet toimivat samalla periaatteella kuin attribuuttien tapauksessa. Niiden lisäksi luokissa määritellään luokkatyyppi sekä vaaditut ja sallitut attribuutit. Luokkatyypeillä voidaan luoda eri tavoin yhdisteltäviä luokkia, joiden avulla saadaan modulaarisesti rakennettuja olioita. [9]

Tiedon välittäminen

LDIF-formaatti (LDAP Data Interchange Format) on tekstitiedostomuoto LDAP-konfiguraatiotiedon ja -hakemistodatan säilyttämiseen. LDIF-tiedostossa voidaan määritellä useita hakemisto-olioita ja niiden attribuutteja. Alkiodatan lisäksi LDIF-tiedostossa on mahdollista määritellä LDAP-muutosoperaatioita [14]. Operaation määrittäminen on tarpeen esimerkiksi, jos olemassa olevaa tietoa halutaan muokata sen sijaan, että luotaisiin uusi olio. Listauksessa 3.1 on esimerkki LDIF-muotoisesta LDAP-oliosta, joka kuvaa yhtä henkilöä. [3]

Listaus 3.1: Henkilöä kuvaava LDAP-olio LDIF-muodossa. [9]

```
dn: cn=John Doe,dc=example,dc=com
objectClass: person
cn: John Doe
sn: Doe
```

DSML:ää (Directory Services Markup Language) voidaan käyttää hakemistodatan siirtämiseen ja käsittelyyn XML:n avulla. DSMLv1 on XML-pohjainen esitysta-

pa hakemistodatalle. Sen tarkoituksena on tuoda hakemistodata XML-sovellusten saataville. [22]

DSMLv2:lla voidaan puolestaan ilmaista hakemisto-operaatioita DSMLv1:n rajoittuessa ainoastaan datan kuvaamiseen. DSMLv2 tukee muutosten lisäksi myös hakuja, vertailua sekä laajennettua operaatiota (Extended Operation). DSMLv2 ei siis kata DSMLv1:n tarjoamia ominaisuuksia. [39]

XML:ään pohjautuvana kielenä DSML soveltuu yleiskäytännölliseksi siirtoformaatiksi. Koska monet sovellukset pystyvät nykyään luomaan ja käsittelemään XML-dataa, on erilaisten palvelujen integroiminen LDAP-hakemistoon DSML:llä vaivattomampaa kuin LDIFillä. Useat yleissovellukset ja ohjelmointikielet tukevat XML-syntaksia, kun taas LDIF on käytössä pääosin vain LDAP-hakemistopalvelimissa. [9]

DSML:llä voidaan kuvata myös hakemistoskeemoja. Tarvittaessa skeemat voidaan liittää siirrettävän hakemostodatan mukaan, jolloin vastaanottaja voi täyttää mahdollisesti puutteellisia skeemojaan. [9]

Monet LDAP-palvelimet eivät kuitenkaan tue DSML-formaattia oletuksena, vaan tuki täytyy lisätä käyttämällä esimerkiksi jotakin väliohjelmistoa. Lisäksi DSML-dokumentit ovat LDIF-tiedostoja monimutkaisempia kirjoittaa sekä lukea ja kooltaan suurempia. Listauksessa 3.2 näkyy DSML-muotoinen yhtä henkilöä kuvaava LDAP-hakemisto-olio. [9]

Listaus 3.2: Henkilöä kuvaava LDAP-olio DSML-muodossa. [9]

```
<dsml:dsml xmlns:dsml="http://www.dsml.org/DSML">
  <dsml:directory-entries>
    <dsml:entry dn="cn=John Doe,dc=example,dc=com">
      <dsml:objectclass>
        <dsml:oc-value>person</dsml:oc-value>
      </dsml:objectclass>
      <dsml:attr name="cn">
        <dsml:value>John Doe</dsml:value>
      </dsml:attr>
      <dsml:attr name="sn">
        <dsml:value>Doe</dsml:value>
      </dsml:attr>
    </dsml:entry>
  </dsml:directory-entries>
</dsml:dsml>
```

[22] Tauber, J., et al. Directory Services Markup Language (DSML). `http://xml.coverpages.org/dsmlv1.html`. Viitattu 26.4.2013.

3.2 Osoitepalveluohjelmistot

Eri protokollilla viestivien ohjelmistojen määrä vaihtelee merkittävästi protokollan levinneisyyden mukaan. Tässä osiossa esitellään osoitepalveluiden pohjana käytettyjä ohjelmistoja. Pääpaino on LDAP-palvelinohjelmistoilla niiden laajan tarjonnan ja käytön vuoksi.

3.2.1 ACAP-, CardDAV- ja X.500-ohjelmistot

ACAPia tukevia sovelluksia on olemassa hyvin vähän. Koska protokolla ei levinnyt laajaan käyttöön, on sitä tukevia palvelintoteutuksia erittäin rajatusti. Jotkin olemassa olevista toteutuksista voivat olla vanhentuneita ja keskeneräisiä. [10]

CardDAV-tuki on yleistynyt lisäominaisuutena erilaisissa palvelinohjelmistoissa, mutta tämän lisäksi on olemassa myös erikseen CardDAVia varten kehitettyjä toteutuksia [25]. Yleistymistä on saattanut helpottaa HTTP-pohjaisuus sekä vCard-formaatin käyttö, sillä näitä teknologioita on käytetty muissa yhteyksissä jo pidempään. Monet CardDAV-palvelimet ovat avointa lähdekoodia, mutta myös joitakin kaupallisia toteutuksia on olemassa [25].

Palvelinten lisäksi on kehitetty myös useita CardDAV-asiakasohjelmia. Ohjelmistokehitystä varten on olemassa joitakin vCard-datan käsittelyyn tarkoitettuja kirjastoja. [23] [24]

LDAP-ratkaisujen yleisyydestä huolimatta X.500-standardien mukaisia hakemistopalvelimia kehitetään edelleen. Useat palvelinohjelmistoista tarjoavat DAP-tuen lisäksi mahdollisuuden myös LDAP-yhteyksien luomiseen. [27]

---

[10] Melnikov, A. ACAP server implementations. `http://www.melnikov.ca/mel/devel/Acap_Server_Ref.html`, 2003. Viitattu 14.2.2013.

[25] The Calendaring and Scheduling Consortium. CardDAV Servers. `http://carddav.calconnect.org/implementations/servers.html`, 2012. Viitattu 14.2.2013.

[23] The Calendaring and Scheduling Consortium. CardDAV Clients. `http://carddav.calconnect.org/implementations/clients.html`, 2012. Viitattu 14.2.2013.

[24] The Calendaring and Scheduling Consortium. CardDAV Libraries & Tools. `http://carddav.calconnect.org/implementations/librariestools.html`, 2012. Viitattu 14.2.2013.

[27] X500Standard.com. X.500 Products by Vendor. `http://www.x500standard.com/index.php?n=Participate.Vendors`. Viitattu 19.4.2013.

3.2.2 LDAP

Useat eri yritykset tarjoavat LDAP-palvelimia organisaatioiden ja tietojärjestelmien hallintaa varten. Esimerkiksi Microsoftilla, Oraclella ja IBM:llä on omat toteutuksensa LDAP-palvelimista. Näiden lisäksi on kuitenkin saatavilla myös ilmaisia ja avoimia ohjelmistoja. Näihin lukeutuvat muun muassa OpenLDAP sekä ApacheDS. Alla on esitelty merkittävimpiä vaihtoehtoja LDAP-pohjaisen tietojärjestelmän rakentamiseen.

389 Directory Server

389 Directory Server on avoimen lähdekoodin LDAP-palvelin, joka käytti aikaisemmin nimeä Fedora Directory Server. 389-projekti käynnistyi vuonna 2005, mutta se rakentuu vanhemman työn pohjalle. Palvelinohjelmisto pohjautuu samaan slapd-projektiin (Stand-alone LDAP Daemon), josta myös OpenLDAP sai alkunsa. 389 Directory Server on suunnattu Linuxille mutta on käännettävissä myös muille alustoille. Ohjelmisto käyttää tiedon säilytykseen Oraclen Berkeley Databasea. Se tukee monen pääkopion replikointimenetelmää sekä tarjoaa DSML-rajapinnan. [3] [2] [1]

Active Directory

Active Directory on Microsoftin kehittämä LDAP-palvelinohjelmisto. Se säilyttää hierarkkisesti tietoja yritysverkon laitteista kuten palvelimista, verkkolevyistä sekä tulostimista. Lisäksi hakemistossa pidetään käyttäjätilejä, sovellus- ja palvelutietoja sekä tietoturvapolitiikkaa koskevia sääntöjä. Käyttäjätietoihin kuuluvat esimerkiksi nimi, sähköpostiosoite ja puhelinnumero. Osoitekirjatoiminnallisuus on siis vain osa Active Directoryn tarjoamista palveluista. Palvelun päätehtävä on hallita verkon resursseja. Tietojen hajauttamiseen Active Directory käyttää monen pääkopion replikointia. [11]

ADAM eli Active Directory Application Mode on Active Directoryyn pohjautuva LDAP-hakemistopalvelin. Nämä ohjelmistot ovat hyvin samanlaisia, mutta ADAM

---

[3] 389 Directory Server. Features. `http://directory.fedoraproject.org/wiki/Features`. Viitattu 26.2.2013.

[2] 389 Directory Server. FAQ. `http://directory.fedoraproject.org/wiki/FAQ`. Viitattu 26.2.2013.

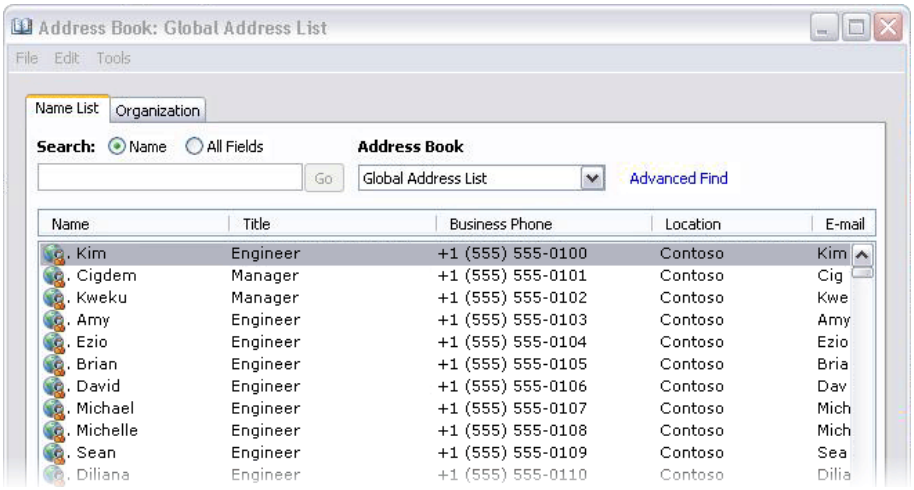
[1] 389 Directory Server. 389 Change FAQ. `http://directory.fedoraproject.org/wiki/389_Change_FAQ`. Viitattu 22.5.2013.

[11] Microsoft TechNet. Active Directory Architecture. `http://technet.microsoft.com/en-us/library/bb727030.aspx`. Viitattu 19.2.2013.

on tarkoitettu palvelinkäyttöjärjestelmien sijaan sovelluksia varten. Active Directorystä poiketen ADAMia voi ajaa Windows XP -käyttöjärjestelmällä. Tämän lisäksi ohjelmistot eroavat tietorakenteidensa osalta. ADAMin ei edellytetä noudattavan samoja rakenteellisia vaatimuksia, jotka Active Directoryn hakemistolle on asetettu. Tämä koskee muun muassa hakemistoskeeman määrittelyitä sekä nimeämiskäytäntöjä. [16]

AD LDS eli Active Directory Lightweight Directory Services on ADAMin pohjalta kehitetty versio Active Directorystä. Sen tarkoitus on ollut korvata ADAM. AD LDS on kohdistettu XP:tä uudemmille Windows-käyttöjärjestelmille. [12]

Microsoftin Exchange-palvelinohjelmisto säilyttää Active Directoryssä GAL-osoitelistaa (Global Address List). GAL sisältää tiedot muun muassa organisaation ryhmistä, käyttäjistä ja kontakteista. Osoitekirjaa voidaan käyttää Microsoftin Outlook-sähköpostiohjelman avulla, ja tiedoista on mahdollista koostaa paikallisia yhteydettömiä osoitelistoja. GAL on näkyvissä kuvassa 3.7. [14] [15]



Kuva 3.7: GAL käyttäjän näkökulmasta. [15]

---

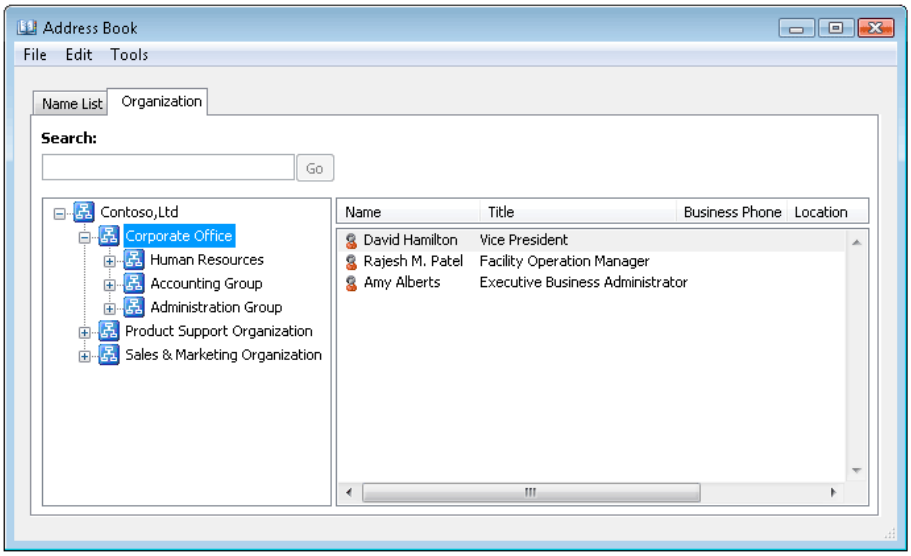
[16] Microsoft TechNet. What Is Active Directory Application Mode? `http://technet.microsoft.com/en-us/library/cc738377(WS.10).aspx`. Viitattu 19.2.2013.

[12] Microsoft TechNet. Active Directory Lightweight Directory Services Overview. `http://technet.microsoft.com/en-us/library/cc754361(WS.10).aspx`. Viitattu 19.2.2013.

[14] Microsoft TechNet. Managing Global Address Lists. `http://technet.microsoft.com/en-us/library/bb232101(EXCHG.80).aspx`. Viitattu 19.2.2013.

[15] Microsoft TechNet. Understanding Address Lists. `http://technet.microsoft.com/en-us/library/bb232119(EXCHG.80).aspx`. Viitattu 19.2.2013.

Uudemmassa Exchange Server 2013:ssa osoitekirjatoiminnallisuutta on parannettu lisäämällä palvelimeen HAB eli Hierarchical Address Book. GAL tarjoaa ainoastaan listan kontakteista, kun taas HABiin voi sisällyttää organisaatiohierarkian. Hierarkian tallentaminen mahdollistaa organisaation henkilö- ja ryhmärakenteen selaamisen sekä helpottaa vastaanottajien löytämistä. HAB on näkyvissä kuvassa 3.8. [13]



Kuva 3.8: HAB käyttäjän näkökulmasta. [13]

ApacheDS

ApacheDS eli Apache Directory Server on kokonaan Javalla toteutettu avoimen lähdekoodin LDAP-palvelin, jonka kehitys aloitettiin vuonna 2002. Se on saatavilla Windowsille, OS X:lle sekä Linuxille. Palvelinta on mahdollista ajaa itsenäisesti tai se voidaan sisällyttää toiseen Java-ohjelmaan. Lisäksi sen arkkitehtuuri mahdollistaa erilaisten laajennuksien tekemisen ja käyttämisen. [4] [5]

---

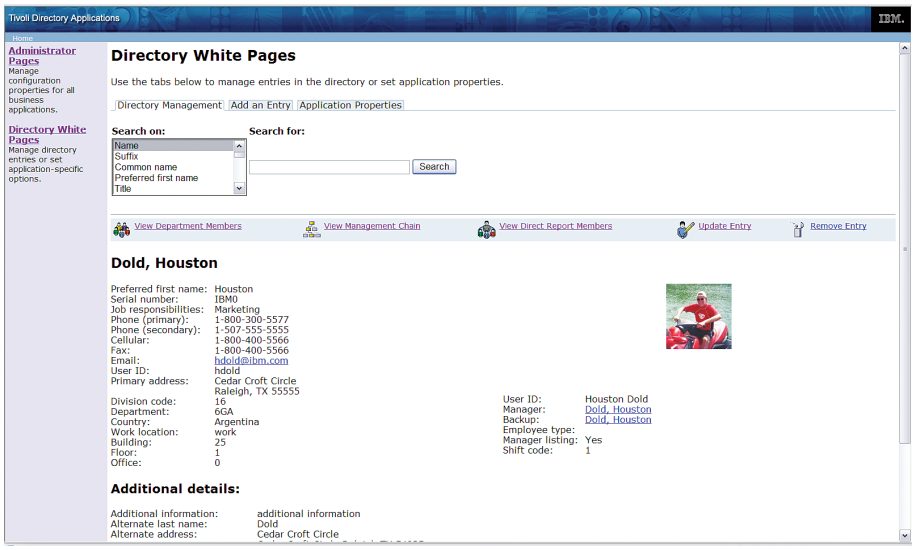
[13] Microsoft TechNet. Hierarchical Address Books. `http://technet.microsoft.com/en-us/library/ff629379.aspx`. Viitattu 19.2.2013.

[4] Apache Directory. ApacheDS v2.0 Basic User's Guide: 1.1 - What Apache Directory Server is. `http://directory.apache.org/apacheds/basic-ug/1.1-what-apacheds-is.html`. Viitattu 27.2.2013.

[5] Apache Directory. Downloads. `http://directory.apache.org/apacheds/downloads.html`. Viitattu 27.2.2013.

IBM Tivoli Directory Server

IBM:n kehittämä LDAP-palvelinratkaisu on nimeltään IBM Tivoli Directory Server. Se on saatavilla useille eri käyttöjärjestelmille kuten Windowsille, Linuxille ja Solarisille. IBM Tivoli Directory Server tarjoaa yritysympäristöihin identiteetinhallintaa ja mahdollistaa muun muassa osoitekirjapalvelun toteuttamisen. Osoitepalvelun web-käyttöliittymä näkyy kuvassa 3.9. [16]



Kuva 3.9: IBM Tivoli Directory Serverin osoitekirjan web-käyttöliittymä [16]

Novell eDirectory

eDirectory on Novellin kehittämä LDAP-ohjelmisto. Se toimii muun muassa Linuxilla ja Windows-palvelimilla. LDAPin lisäksi se tukee myös DSML:ää SOAPin välityksellä. Hakemistoa voidaan käyttää säilyttämään ja jakelemaan tietoa esimerkiksi palvelimista, käyttäjistä ja verkon resursseista. [17] [18]

[17] NOVELL. Develop to eDirectory. `http://www.novell.com/developer/develop_to_edirectory.html`. Viitattu 27.2.2013.

[18] NOVELL. Glossary: eDirectory. `http://www.novell.com/communities/glossary/term/3276`. Viitattu 27.2.2013.

OpenDS ja OpenDJ

OpenDS on ilmainen, avoimeen lähdekoodiin pohjautuva hakemistopalvelinohjelmisto. Sen kehitys alkoi vuonna 2005. OpenDS tarjoaa LDAP-rajapinnan lisäksi myös DSML-tuen. Alun perin projektin kehittäjänä toimi Sun Microsystems, mutta myöhemmin ohjelmisto siirtyi Oraclen alaisuuteen. Varsinainen kehitystyö kuitenkin tapahtuu vapaaehtoisyhteisön voimin. Oracle kehittää ohjelmistosta omaa kaupallista jakeluversiotaan, Oracle Unified Directoryä [31]. OpenDS on tehty kokonaan Javalla, mikä mahdollistaa laajan käyttöjärjestelmätuen. Hakemiston on testattu toimivan muun muassa Windows- ja Solaris-palvelimilla sekä eri Linux-jakeluilla. [9] [8]

OpenDS ei ole enää aktiivisessa kehityksessä, mutta ohjelmistosta on tehty kehityshaara nimeltä OpenDJ. OpenDJ on OpenDS:lle pohjautuva hakemistopalvelin, johon on tehty useita parannuksia ja korjauksia. Myös OpenDJ perustuu avoimelle lähdekoodille, on ilmainen sekä toimii Linuxilla ja Windows-palvelimilla. Kehitystä ohjaa ForgeRock. [5] [7]

OpenLDAP

OpenLDAP on ilmainen LDAP-palvelin. Projekti aloitettiin vuonna 1998, mutta palvelin pohjautuu vanhempaan hakemisto-ohjelmistoon. OpenLDAP perustuu avoimeen lähdekoodiin ja on toteutettu C-kielellä. Ohjelmistoa on mahdollista ajaa monilla eri käyttöjärjestelmillä, joihin lukeutuvat muun muassa eri ja Windows-, Linux- ja Unix-versiot. [19] [21] [20]

OpenLDAPin taustaosaksi on tarjolla useita eri vaihtoehtoja. Monet taustaosista keskittyvät tiedon varastoimiseen, mutta niitä voidaan käyttää myös muihin tarkoituksiin. Taustaosilla voidaan muun muassa tehdä välityspalvelintoiminnallisuutta, SQL-rajapinta (Structured Query Language) tai integraatio Perl-ohjelmaan. Lisäksi OpenLDAP tukee päällysosia. Niillä voidaan muuttaa palvelimen toimin-

---

[9] Java.net. OpenDS FAQ – Project Definition. `http://java.net/projects/opends/pages/2_4_ProjectDefinition`. Viitattu 26.2.2013.

[8] Java.net. OpenDS FAQ – OpenDS System Requirements. `http://java.net/projects/opends/pages/2_4_OpenDSSystemRequirements`. Viitattu 26.2.2013.

[7] ForgeRock Community. OpenDJ Directory Services Project: FAQ. `http://opendj.forgerock.org/faq.html`. Viitattu 26.2.2013.

[19] OpenLDAP. Main Page. `http://www.openldap.org/`. Viitattu 28.2.2013.

[21] OpenLDAP. Programming Guidelines. `http://www.openldap.org/devel/programming.html`. Viitattu 28.2.2013.

[20] OpenLDAP. OpenLDAP Faq-O-Matic: Is OpenLDAP Software for Linux only? `http://www.openldap.org/faq/data/cache/1142.html`. Viitattu 28.2.2013.

taa tekemättä muutoksia taustaosaan. Päällysosilla voidaan esimerkiksi toteuttaa pääsynvalvonta, LDAP-kyselyiden uudelleenkirjoitus ja uniikkien attribuuttiarvojen takaaminen. [23]

OpenLDAP mahdollistaa myös erilaisten replikointitopologioiden rakentamisen. Niihin kuuluu esimerkiksi monen pääkopion replikointi. Pääsynhallinnan määrittämisessä voidaan käyttää apuna muun muassa hakemistohierarkiaa, regex-lausekkeita (regular expression) sekä ryhmämäärittelyitä. [23]

Oraclen hakemistopalvelimet

Oraclen valikoimassa on kolme erilaista hakemistopalvelinta. Näistä kukin on saatavilla useille eri alustoille, joihin lukeutuvat muun muassa eri Linux-jakelut ja Windows-käyttöjärjestelmäversiot. [26] [28] [30]

Oracle Directory Server Enterprise Edition (ODSEE) on Oraclen tarjoama suuriin yritysympäristöihin suunniteltu hakemistopalvelin. Alun perin kehittäjänä toimi Sun, jolloin ohjelmiston nimi oli Sun Directory Server Enterprise Edition. ODSEE tukee datan replikointia ja mahdollistaa pääsynhallintasääntöjen määrittämisen. Lisäksi se sisältää välityspalvelimen, joka tekee kuormantasausta ja parantaa tiedon saatavuutta. [27]

Oracle Internet Directory (OID) on yleiskäyttöinen LDAP-palvelin. Oracle-ympäristöissä sitä käytetään käyttäjätietojen säilyttämiseen identiteentinhallintajärjestelmässä. OID voidaan integroida useisiin eri palveluihin, joilla hallitaan käyttäjiä. Näihin palveluihin lukeutuvat muun muassa Active Directory ja OpenLDAP. [29]

Oracle Unified Directory (OUD) perustuu OpenDS-projektiin. OpenDS:n tavoin OUD on tehty Javalla. OUD käyttää Oraclen Berkeley Database -tietokantaa. [31]

Yhteenveto

LDAP-palvelimia on saatavilla runsaasti yleisimmille alustoille. Lähes kaikkia merkittävimmistä ohjelmistoista on mahdollista ajaa ainakin jollakin Linux-jakelulla tai Windows-versiolla. Sekä kaupalliset että avoimet ja ilmaiset palvelintoteutukset tarjoavat olennaisimmat toiminnut, mutta yksityiskohtaisemmat ominaisuudet saattavat vaihdella. Ohjelmistoa valittaessa onkin täten arvoitava näiden ominaisuuksien tarpeellisuutta. Lisäksi valintaan vaikuttavat esimerkiksi palvelimen laajennettavuus lisäosilla ja teknisen tuen laajuus. Taulukossa 3.1 näkyy yleiskatsaus esitellyistä LDAP-palvelimista.

Taulukko 3.1: Merkittävimpien LDAP-hakemistopalvelinten vertailu.

Palvelinohjelmisto	**Avoin**	**Ilmainen**	**Linux**[1]	**Windows**[1]
389 Directory Server	Kyllä	Kyllä	Kyllä	Kyllä
Active Directory	Ei	Ei[2]	Ei	Kyllä
AD LDS	Ei	Ei[2]	Ei	Kyllä
ADAM	Ei	Ei[2]	Ei	Kyllä
ApacheDS	Kyllä	Kyllä	Kyllä	Kyllä
IBM Tivoli Directory Server	Ei	Ei	Kyllä	Kyllä
Novell eDirectory	Ei	Ei	Kyllä	Kyllä
OpenDJ	Kyllä	Kyllä	Kyllä	Kyllä
OpenDS	Kyllä	Kyllä	Kyllä	Kyllä
OpenLDAP	Kyllä	Kyllä	Kyllä	Kyllä
Oracle Directory Server EE	Ei	Ei	Kyllä	Kyllä
Oracle Internet Directory	Ei	Ei	Kyllä	Kyllä
Oracle Unified Directory	Ei	Ei	Kyllä	Kyllä

[1]Yksi tai useampi käyttöjärjestelmäversio tai -jakelu.
[2]Vaatii maksullisen käyttöjärjestelmän.

Luku 4

Tietomallit ja tietoliikenne

Tässä luvussa esitellään erilaisia tietomalleja sekä tiedon replikointiin liittyviä konsepteja. Tietomalli vaikuttaa olennaisesti osoitepalvelun tietorakenteisiin ja siihen, miten tietoa voidaan käsitellä. Oikeilla replikointivalinnoilla puolestaan varmistetaan ajantasaisen tiedon saatavuus haasteellisessakin verkossa.

4.1 Tietomallit

Tässä osiossa käydään läpi yleisimpiä tietokantajärjestelmissä käytettyjä tietomalleja. Tietomallilla tarkoitetaan dataa kuvaavaa merkintätapaa sekä operaatioita, joilla kyseistaä dataa käsitellään [40]. Sopivan tietomallin valinta on olennaista, jotta osoitekirjan tiedot saadaan järjestettyä niille luonnollisesti sopivaan muotoon. Tämä parantaa tiedon saatavuutta ja helpottaa pääsynhallinnan toteuttamista.

Relaatiomalli

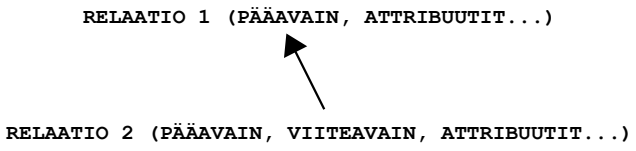
Relaatiomalli on yleisin nykysovelluksissa käytetty tietokantamalli, ja tarjolla onkin useita relaatiotietokannan hallintajärjestelmiä [15]. Relaatiotietokanta koostuu nimensä mukaisesti relaatioista, joiden tarkoitus on kuvata joukkoa tallennettavia attribuutteja. Yksi tapa esittää relaatio on muotoa `KAUPUNKITIEDOT(KAUPUNKI, OSAVALTIO,ASUKASLUKU)`. Tällaisella relaatiolla voidaan kuvata kaupunkien tietoja, missä attribuutteina ovat kaupungin nimi, osavaltio sekä asukasluku. Kyseisellä relaatioilla kuvattu data voidaan esittää taulumuodossa kuvan 4.1 tavoin. Täten kullakin attribuutilla on pystysarake ja tietueet ovat vaakariveillä. [40]

Kaupunki	Osavaltio	Asukasluku
San Diego	Texas	4490
Miami	Oklahoma	13880
Pittsburg	Iowa	509

Kuva 4.1: Esimerkki relaatiotietokannan taulusta. [40]

Relaatiomallin tietokannoissa dataa käsitellään relaatioalgebrassa määritellyillä operaatioilla. Näillä operaatiolla voidaan muun muassa valita joukosta ainoastaan halutut ehdot täyttävät tietueet tai suorittaa liitos eri relaatioiden kesken. [40]

Tietueet erotellaan toisistaan käyttämällä niille uniikkeja avaimia. Avaimet voivat koostua yhdestä tai useammasta attribuutista, joiden arvojen yhdistelmä on yksilöllinen. Näiden avainten ja valittujen tietokantaoperaatioiden avulla on mahdollista muodostaa yhteyksiä eri taulujen välille. Yhteydet voivat olla tyypiltään esimerkiksi yhden-suhde-yhteen tai monen-suhde-moneen. Kuva 4.2 esittää relaatioiden välistä viittausta relaatiotietokannassa. [40]



Kuva 4.2: Viittaus relaatiotietokannassa. [15]

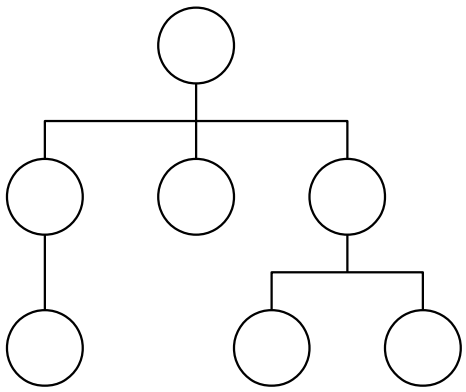
Hierarkkinen malli

Puulla tarkoitetaan hierarkkista tietorakennetta, joka koostuu solmuista ja niitä yhdistävistä kaarista. Tietojenkäsittelyssä puu piirretään ylösalaisin, ja ylintä solmua sanotaan juureksi. Kunkin solmun välittömästi alapuolella olevia solmuja sanotaan sen lapsiksi ja kaikkia alla olevia solmuja jälkeläisiksi. Vastaavasti välittömästi yläpuolella olevaa solmua sanotaan vanhemmaksi ja kaikkia yläpuolella olevia solmuja esi-isiksi. Saman vanhemman lapsia sanotaan sisaruksiksi. [18]

Sisäsolmulla tarkoitetaan solmua, jolla on yksi tai useampi lapsi, kun taas lehtisolmu on lapseton. Polku on kahden solmun välissä olevien kaarien muodostama reitti. Tasolla tai syvyydellä tarkoitetaan juuren ja solmun välisen polun pituutta. Alipuu on valittuun solmuun juurtuva puurakenne. [18]

Hierarkkisessa tietokantamallissa tieto järjestellään puutietorakenteeseen. Tämä malli soveltuu tilanteisiin, joissa tallennettavan datan rakenne järjestyy luonnostaan hierarkkisesti. Mallia on käytetty tietokannoissa jo pitkään ja se on nykypäivänä yleinen suurissa yrityksissä. [15]

Hierarkkisessa tietokannassa tietueet ovat puiden solmuja, jotka yhdistetään linkeillä. Näillä oliomaisilla solmuilla on kenttiä, joihin voidaan tallentaa arvoja. Tietueet erotetaan toisistaan uniikeilla tunnuksilla, joita voidaan pitää tietueiden osoitteina. Hierarkkisessa mallissa operoidaan tyypillisesti yksisuuntaisilla linkeillä. Toisin sanoen linkkejä pitkin voidaan kulkea vain lapsisolmujen suuntaan. Muihin solmuihin on kuitenkin mahdollista päästä juurisolmusta käsin tai käyttäen virtuaalisia tietueita, mikäli sellaisia on mahdollista luoda. Hierarkkinen tietokantamalli on esitetty kuvassa 4.3. [40]



Kuva 4.3: Hierarkkinen tietokantamalli. [15]

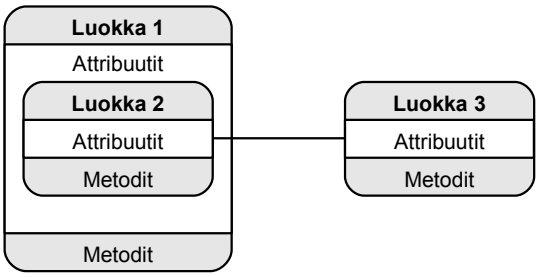
Oliomalli

Oliopohjaisissa tietokannoissa tiedot tallennetaan luokkamääritelmille perustuviin olioihin. Luokat koostuvat erilaisista tietorakenteista, joihin varsinainen data säilötään, sekä metodeista, joilla kyseistä dataa voidaan manipuloida. Oliopohjaisilla tietokannoilla on muutama yhteinen ominaispiirre. Ensinnäkin tallennettavilla olioilla on oma oliotunnus eli uniikki osoite. Toisekseen nämä kannat mahdollistavat uusien, monimutkaisten oliotyyppien luomisen. Näiden lisäksi oliotietokannat hyödyntävät tyyppihierarkiaa, jonka avulla voidaan luoda alityyppejä.

Oliot voivat koostua perustietotyypeistä, joukoista sekä viittauksista muihin olioihin. Tyyppimääritelmä voidaan esittää muodossa `AsiakasTyyppi = RECORDOF( nimi: string, osoite: string, saldo: int, tilaukset: SETOF(Tilaus-`

`Tyyppi))`. Tämä tyyppi kuvaa asiakasta ja sen tilauksia, ja sille on määritelty merkkijonoja, kokonaisluku sekä joukko muita olioita. [40]

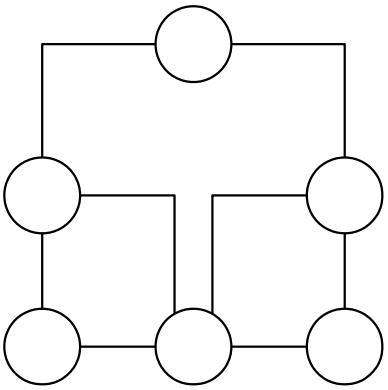
Oliotietojen käsittely tapahtuu luokissa kuvattujen metodien avulla. Yksinkertainen merkkijonon palauttava metodi voidaan esittää esimerkiksi muodossa `HaeNimi: return(nimi)`. Luokat voivat myös periytyä toisistaan, minkä seurauksena on mahdollista luoda aliluokkia, joiden toiminnallisuutta on laajennettu tai mukautettu. Oliomallia on hahmoteltu kuvassa 4.4. [40]



Kuva 4.4: Esimerkki oliomallisen tietokannan luokkamäärittelystä. [15]

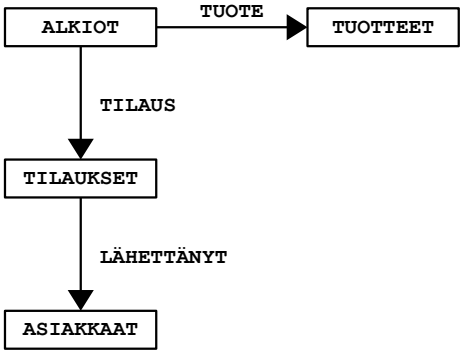
Verkkomalli

Verkkomallisessa tietokannassa kunkin tietueen voi yhdistää mielivaltaiseen määrään muita tietueita. Tällainen tietokantamalli on hyvin joustava ja mahdollistaa käytännössä minkälaisen tahansa suhteen muodostamisen. Verkkotietokannat vaativat merkittävissä määrin suunnittelua, mutta niiden avulla voidaan rakentaa hyvin optimoituja järjestelmiä. Useimmat verkkomalliset tietokannat toimivat yhden-suhde-moneen -yhteyksillä mutta myös monen-suhde-moneen -yhteydet ovat käytössä joissakin toteutuksissa. Verkkomalli näkyy kuvassa 4.5. [15]



Kuva 4.5: Verkkomallinen tietokantarakenne olioineen ja linkkeineen. [15]

Tietueet verkkomallisissa tietokannoissa ovat olioita. Ne koostuvat kentistä, joihin voidaan tallentaa erilaisissa perustietotyypeissä olevia arvoja. Oliopohjaisuudesta johtuen tietueet erotetaan toisistaan sisäisellä oliotunnuksella. Käyttämällä näitä yksilöllisiä tunnisteita osoittimina voidaan olioiden välille rakentaa linkeiksi kutsuttuja yhteyksiä. Kuvassa 4.6 näkyy tilausjärjestelmää kuvastava verkkomalli, joka koostuu asiakkaista, tilauksista ja tuotteista. Relaatioalgebran liitosoperaatiota vastaavia toimenpiteitä voidaan suorittaa seuraamalla linkkejä verkossa. Esitetyn tapauksen asiakkaita ja heidän tilaamiaan tuotteita kuvaava linkinseuraamisoperaatio voidaan ilmaista muodossa `LÄHETTÄNYT(TILAUS(TUOTE(TUOTTEET)))`. [40]

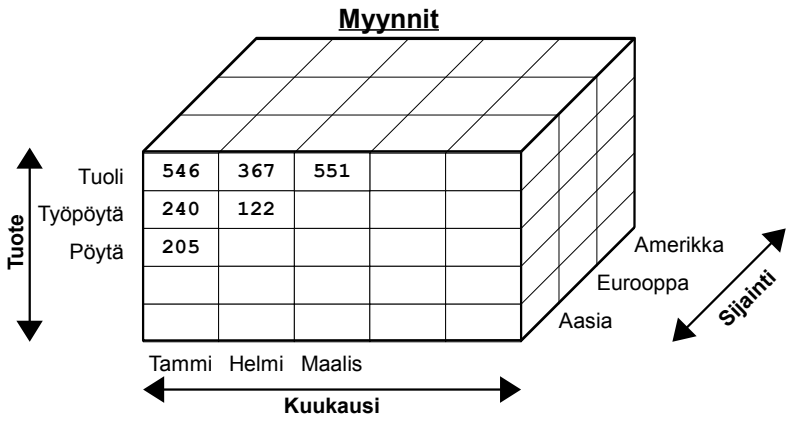


Kuva 4.6: Yksinkertaista tilausjärjestelmää esittävä rakenne. [40]

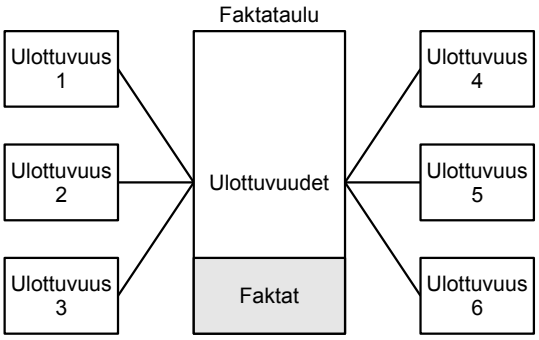
Moniulotteinen malli

Moniulotteista tietokantamallia suositaan silloin, kun tarkoituksena on suurien datamäärien analysointi. Mallia käytettään usein suurissa tietovarastosovelluksissa, verkkopohjaisissa data-analyysiohjelmissa ja tiedonlouhinnassa. [32]

Moniulotteinen malli voidaan visualisoida kahdella tavalla: kuutiona tai tähtiskeemana. Kuutionäkymässä tieto tallennetaan soluihin, joihin osoitetaan kuution sivujen määrittämillä ulottuvuuksilla. Ulottuvuudet määräävät, miten tieto kuutiossa on jaoteltu. Jaottelu voi tapahtua esimerkiksi aikavälin, sijainnin ja tuotteen mukaan. Solussa on tällöin näiden parametrien leikkauskohdalle relevanttia tietoa. Tähtiskeemanäkymässä keskelle piirretään yhden solun esitys. Sen ympärille kuvataan kaikki solun määrittävät ulottuvuudet. Keskiruudussa on kaikkien näiden ulottuvuuksien määrittämä yhdistetty avain ja kyseistä avainta vastaavan solun sisältämät tiedot. Kuvassa 4.7 näkyy moniulotteisen mallinen kuutionäkymä ja kuvassa 4.8 tähtiskeemanäkymä. [15]



Kuva 4.7: Moniulotteisen tietokantamallin kuutioesitys. [15]



Kuva 4.8: Moniulotteisen tietokantamallin tähtiskeemaesitys. [15]

Tiedon jäsentely ulottuvuuksiin mahdollistaa tiettyjen mallille ominaisten dataoperaatioiden suorittamisen. Näihin kuuluvat muun muassa kuution pilkkominen ja pyörittäminen. Tiedolle voidaan myös tehdä aggregointia tai relaatiomallille tyypillisiä liitosoperaatioita. [32]

Yhdistetty relaatiokanta ja hierarkkinen malli

Tarvittaessa relaatiotietokanta ja hierarkkinen malli voidaan yhdistää. Tässä ratkaisussa hierarkkisella mallilla toimiva hakemistopalvelin käyttää tiedon varastointiin relaatiomallista taustaosaa. Järjestely mahdollistaisi tiedon saatavuuden sekä hakemisto- että relaatiorajapinnan kautta. Tietomallien eroavaisuuksien vuoksi varsinainen hyöty useamman rajapinnan käytöstä jäisi kuitenkin varsin pieneksi. [23]

Yksi tapa yhdistää mallit on hajauttaa hakemisto-oliot eri tauluihin. Hajautus on tehtävä moniarvoisten attribuuttien ja mahdollisten useiden olioluokkien vuoksi. Tietojen käsittely relaatiorajapinnan kautta olisi kuitenkin varsin työlästä tietoolioiden pirstaloitumisen takia. Tiedon haku erityisesti suodattimia käytettäessä hidastuu, koska data on jaettu useaan eri paikkaan. Toisessa ratkaisussa hakemisto-oliot tallennetaan binääriolioina relaatiotietokantaan. Tällöin kuitenkaan SQL-hakuja ei voida suorittaa mielekkäästi, joten kyseinen rajapinta jää käyttökelvottomaksi. [23]

4.2 Replikointi

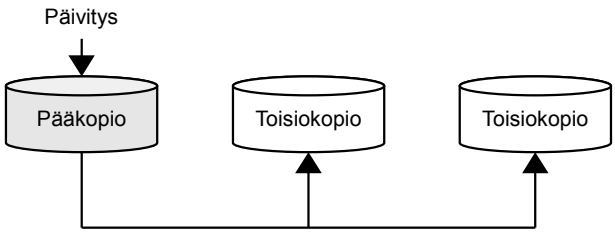
Tässä osiossa käydään läpi erilaisia tiedon replikoinnissa käytettyjä menetelmiä. Tarkoituksena on kartoittaa eri vaihtoehtojen etuja ja haittoja. Osio käsittelee verkkoarkkitehtuurin ja tietoliikenteen lisäksi dataristiriitojen ratkaisua.

4.2.1 Replikointitekniikat

Yhden ja monen pääkopion replikointi

Yhden pääkopion replikoinnissa ainoastaan yhdellä laitteella on täysi hallinta säilytettävään dataan. Pääkopiolla on sekä luku- että kirjoitusoikeudet tietoihin, kun taas muilla kopioilla on ainoastaan lukuoikeudet. Pääkopio käsittelee kaikki hakemistoon tehtävät muutokset ja välittää ne muille laitteille. Tällä järjestelyllä varmistetaan, että ristiriitaisia päivityksiä ei voida tehdä, joten kaikki kopiot pysyvät yhdenmukaisina. Replikointi voi toimia joko työntö- tai vetomallilla. Työntömallissa pääkopio tekee aloitteen päivitysten levittämisestä, kun taas vetomallissa toisiokopiot pyytävät pääkopiolta muutokset. [34]

Yhden pääkopion replikointijärjestely on altis ongelmille, koska pääkopiolaite toimii nyt pullonkaulana muutosoperaatioille. Lisäksi päälaitteen toimimattomuus estää päivitysten tekemisen koko järjestelmässä. Yhden pääkopion replikointijärjestelmä näkyy kuvassa 4.9. [34]

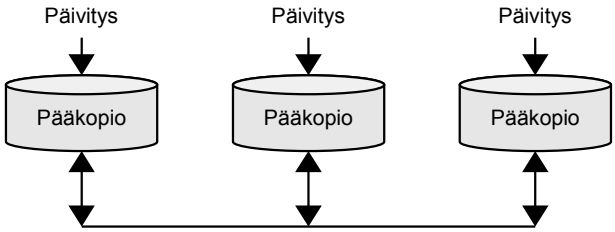


Kuva 4.9: Yhden pääkopion replikointi. [20]

Monen pääkopion replikointijärjestelmässä laitteet ovat tasavertaisia, joten mikä tahansa kopio voi ottaa vastaan muutospäivityksiä. Muutokset on kuitenkin välitettävä kultakin palvelimelta kaikille muille palvelimille. [34]

Monen pääkopion järjestelmällä voidaan välttää yhden pääkopion mallin ongelmat pääpalvelimen suorituskyvystä ja saavuttamattomuudesta. Tietojen muokkaaminen eri laitteilla kuitenkin mahdollistaa ristiriitaisten päivitysten syntymisen. Jotta

tietokannat pysyisivät yhdenmukaisina, täytyy järjestelmän pystyä havaitsemaan ja ratkaisemaan ristiriidat. Monen pääkopion replikointi näkyy kuvassa 4.10. Yhden ja monen pääkopion replikointeja on vertailtu taulukossa 4.1. [34]



Kuva 4.10: Monen pääkopion replikointi. [20]

Taulukko 4.1: Yhden ja monen pääkopion replikoinnin vertailu. [20]

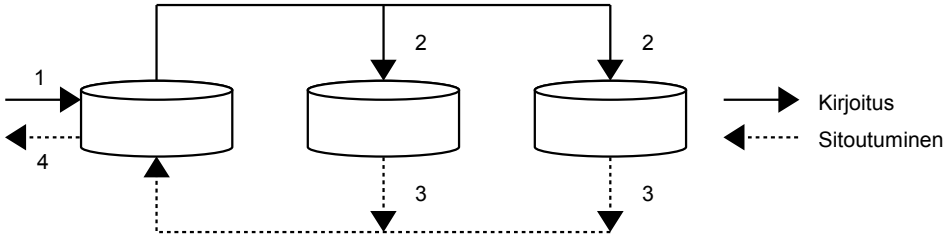
	Yksi pääkopio	**Monta pääkopiota**
Ominaispiirre	Yksi pääkopio	Monta tasavertaista kopiota
Uusimpien tietojen sijainti	Pääkopiolla	Tuntemattomalla kopiolla
Päivitysmekanismi	Keskitetty	Hajautettu
Pullonkaula	Pääkopio	Ei ole

Tahdistettu ja tahdistamaton replikointi

Tietojen päivityksessä käytetään usein käsitteitä transaktio, sitoutuminen ja peruutus. Transaktiolla tarkoitetaan joukkoa kirjoitusoperaatioita ja sitoutumisella transaktion virheetöntä suorittamista. Mikäli päivityksessä tulee virheitä, transaktio peruutetaan. Jotta replikointijärjestelmä toimisi, täytyy kaikki yhdessä paikassa suoritetut transaktiot välittää muille laitteille. Replikoidut transaktiot voidaan suorittaa kahdella periaatteella: tahdistetusti tai tahdistamattomasti. Tahdistetussa replikoinnissa päivityksiä ei suoriteta millään laitteilla ennen kuin kaikki muut laitteet ovat valmiita. Tahdistamattomassa replikoinnissa päivitykset voidaan suorittaa sitä mukaa, kun ne saapuvat. Päivityksien välitystä voidaan tällöin tehdä taustalla. [34]

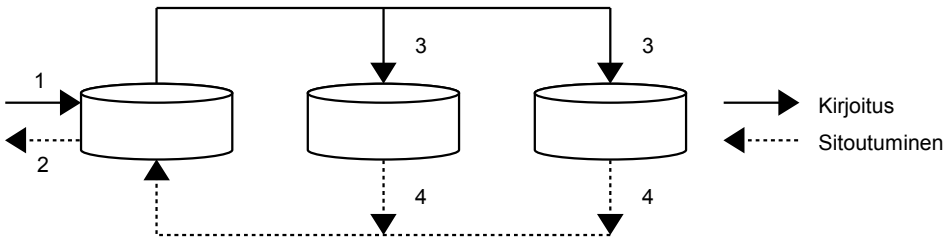
Koska tahdistetussa replikoinnissa transaktiot välitetään kaikille kopioille ja suoritetaan samanaikaisesti, takaa järjestelmä atomisuuden. Toisin sanoen transaktiot sitoutuvat kaikkialla tai eivät missään. Tämä periaate toteutuu esimerkiksi kaksivaiheisessa sitoutumisprotokollassa. Protokollassa yksi osapuoli toimii koordinaattorina, joka transaktion saapuessa pyytää muita osapuolia äänestämään ja

kerää äänet. Mikäli kaikki äänestävät transaktion sitoutumisen puolesta, antaa koordinaattori osallistujille sitoutumisluvan, muutoin koko operaatio peruutetaan. Jos yksikin osapuoli on sitoutumista vastaan, lähetetään puolesta äänestäneille peruutusviesti. Jos jokin osallistujista äänestää sitoutumisen puolesta ja menettää yhteyden koordinaattoriin, jää kyseinen osallistuja estettyyn tilaan. Esto on purettava, jotta normaali toiminta voi jatkua. Näin ollen estävät mekanismit tuovat replikointijärjestelmään merkittäviä suorituskyvyllisiä ja toiminnallisia rajoitteita. Ongelmat vahvistuvat huomattavasti, jos käytössä on haasteellinen verkko. Tahdistettu replikointi on esitetty kuvassa 4.11. [34] [1]



Kuva 4.11: Tahdistettu replikointi. [34]

Tahdistamattomassa replikoinnissa transaktiot voidaan suorittaa kullakin laitteella välittömästi niiden saapuessa. Tällöin eri kopiot voivat ajautua ainakin hetkellisesti erilaisiin tiloihin. Tahdistamattomat replikointimenetelmät voidaan luokitella kahteen eri ryhmään: optimistisiin ja pessimistisiin. Optimistiset mekanismit olettavat, että ristiriitaisia päivityksiä saapuu vain harvoin. Ristiriidan sattuessa virheet pyritään korjaamaan jälkikäteen. Pessimistiset menetelmät puolestaan olettavat ristiriitojen olevan yleisiä ja pyrkivät erilaisilla keinoilla parantamaan eri kopioiden yhdenmukaisuutta. Tahdistamaton replikointi on huomattavasti tehokkaampaa kuin tahdistettu mutta ei takaa kopioiden yhdenmukaisuutta. Replikointimekanismi on esitetty kuvassa 4.12. Tahdistettua ja tahdistamatonta replikointia on vertailtu taulukossa 4.2. [34]



Kuva 4.12: Tahdistamaton replikointi. [34]

Taulukko 4.2: Tahdistetun ja tahdistamattoman replikoinnin vertailu. [20]

	Tahdistettu replikointi	**Tahdistamaton replikointi**
Ominaispiirre	Yksi yhteinen transaktio	Toisistaan riippumattomat sitoutumiset
Yhdenmukaisuus	Täsmällinen	Lopulta saavutettava
Skaalautuvuus	Kymmeniä	Satoja tai jopa enemmän

Täysi ja osittainen replikointi

Täysi replikointi tarkoittaa, että kaikki säilytettävään dataan tehdyt muutokset välitetään kaikille laitteille. Tällöin jokaisella kopiolla on täsmälleen samat tiedot. Tällä järjestelyllä voidaan parantaa tiedon saatavuutta ja viansietokykyä, mutta laitteilta vaaditaan enemmän resursseja kuten muistia. [34]

Osittaisessa replikoinnissa kukin laite säilyttää ainoastaan osajoukkoa kaikesta säilytettävästä tiedosta. Tämä järjestely vähentää laitevaatimuksia ja päivitysliikenteen määrää mutta monimutkaistaa huomattavasti replikointimekanismia. Lisäksi tiedon saatavuus luonnollisesti heikkenee. Täyttä ja osittaista replikointia on vertailtu taulukossa 4.3. [34]

Taulukko 4.3: Täyden ja osittaisen replikoinnin vertailu. [20]

	Täysi replikointi	**Osittainen replikointi**
Ominaispiirre	Kaikki data kaikkialla	Eri kopiolla eri osajoukot
Kuormantasaus	Yksinkertainen	Monimutkainen
Datan saatavuus	Maksimaalinen	Alhaisempi
Käytetyn tilan ja liikenteen määrä	Suuri	Alhaisempi

Tilan ja operaation siirto

Muutosoperaatiota voidaan siirtää kahdessa eri muodossa: tiloina tai operaatioina. Tiloja siirrettäessä muokattava olio lähetetään muille kopioille kokonaisuudessaan. Vastaanottava laite suorittaa tällöin ylikirjoituksen vanhalle oliolle. Menetelmä on yksinkertainen mutta saattaa lisätä tarpeettoman tiedon siirtoa. [36]

Operaatiosiirrossa muutokset välitetään semanttisessa muodossa. Käytännössä lopputilaan siis päästään välittämällä haluttu operaatio parametreineen. Verkkoliikenteen vähentämisen lisäksi operaatioiden siirtäminen mahdollistaa joustavamman ristiriitojen hallintamekanismin. [36]

Yhdenmukaisuustakuu

Replikoinnilla pyritään pitämään eri paikkojen kopiot yhdenmukaisina. Yhdenmukaisuutta voidaan kuitenkin ylläpitää eri tavoilla, jotka puolestaan johtavat eritasoisiin yhdenmukaisuustakuisiin. *Yhden kopion konsistenssi* tarkoittaa, että rinnakkaisoperaatiosarjan ajaminen hajautetussa järjestelmässä tuottaa saman lopputuloksen kuin samojen operaatioiden suorittaminen sarjassa yhdellä laitteella. [36]

Lopulta saavutettavalla konsistenssilla tarkoitetaan, että eri paikkojen kopiot konvergoituvat, kun tarpeeksi aikaa on kulunut. Takeita datan oikeellisuudesta joka hetkellä ei ole, joten joltakin kopiolta tietyllä ajanhetkellä haettu tieto saattaa olla vanhentunutta. Suurin osa optimistisista replikointijärjestelmistä toimii lopulta saavutettavan konsistenssin periaatteella. [36]

Näiden kahden eri ääripään konsistenssitakuiden välillä on olemassa myös muita ratkaisuja. Niissä käytetään tyypillisesti kirjoitusestoja yhdenmukaisuuden edistämiseksi. [36]

4.2.2 Ajastus ja ristiriitojen ratkaiseminen

Syntaktinen ja semanttinen ajastus

Replikointijärjestelmissä ajastuksella tarkoitetaan päivitysten asettamista järjestykseen. Sen tavoitteena on saada kaikki tietokantakopiot samaan tilaan. Syntaktinen ajastus käyttää järjestämiseen ainoastaan tietoa muutoksen ajankohdasta, alkupaikasta ja tekijästä. Esimerkiksi aikaleimojen käyttö on syntaktinen ajastusmekanismi. Semanttisessa ajastuksessa hyödynnetään päivitysoperaatioiden ominaisuuksia kuten suoritusjärjestyksen vaihdannaisuutta ja idempotenssia eli saman

päivityksen uudelleen suorittamista silloin, kun se ei vaikuta lopputilaan. Päivitysjärjestelmä voi esimerkiksi tutkia kahden päivityksen sisältöä ja pyrkiä arvioimaan, onko niiden suoritusjärjestyksellä merkitystä, tai suorittaa replikoitavalle päivitysoperaatiolle konvergenssiin eli yhdenmukaisuuteen ohjaavan muutoksen. Semanttinen järjestelmä voi käyttää avuksi myös tapahtumajärjestystietoa.

Syntaktinen järjestelmä on helpompi toteuttaa mutta voi aiheuttaa virheellisiä esiintymiä. Toisin sanoen päivitysten järjestys voitaisiin nähdä vääräksi, vaikka sillä ei olisikaan vaikutusta tietokannan lopulliseen tilaan. Tästä voi aiheutua tarpeettomia takaisinpaluuoperaatioita. [36]

Tapahtumajärjestyksen määrittäminen

Jotta replikoitavat tietokannat konvergoituisivat, ajastusjärjestelmän on olennaista pystyä määrittämään muutosoperaatioiden tapahtumajärjestys. Verkossa, jossa siirtoviiveet ovat arvaamattomia, voi tietokantapäivitysten järjestys muuttua matkalla. Tästä syystä laitteiden täytyy kyetä määrittämään päivitysten tapahtumajärjestys vastaanottojärjestyksestä riippumatta. Tapahtumajärjestyksen määrittäminen mahdollistaa oikean suoritusjärjestyksen selvittämisen lisäksi myös ristiriitojen havaitsemisen. Järjestyksen määrittämisessä voidaan käyttää apuna erilaisia kellomekanismeja kuten vektorikelloja ja reaaliaikaisia kelloja. [36]

Vektorikellot

Yksi tapa seurata tapahtumajärjestystä on käyttää vektorikelloja. Vektorikello on tyypillisesti jonkinlainen taulukko, jossa avaimina toimivat päivityksiä tekevien tahojen tunnukset ja arvoina aikaleimat. Tunnus voi olla esimerkiksi verkko-osoite ja aikaleimana kasvava kokonaisluku. Päivityksiä levitettäessä laitteet kasvattavat omaa aikaleimaansa ja liittävät uuden vektorikellon viestiin. Liitetty vektorikello sisältää siis myös aikaisempien välittäjien aikaleimat. Vanhentuneet muutokset samaan olioon voidaan nyt erottaa uusista vertaamalla vektorikellojen arvoja. Vektorikellon voidaan sanoa dominoivan toista, mikäli kaikki sen aikaleimat ovat suurempia kuin toisen. Dominoiva kello ilmaisee uudempaa päivitystä. Mikäli muualla verkossa tehdään ristiriitainen muutos, on sillä oma vektorikellonsa. Ristiriitaiset päivitykset voidaan havaita ristiriitaisista vektorikelloista. Toisin sanoen kummassakin kellossa on silloin ainakin yksi aikaleima, joka on toisen kellon vastaavaa uudempi. Tällöin ristiriita on ratkaistava ja voimaan jäävä päivitys voidaan asettaa dominoivaksi yhdistämällä vektorikellot. Vektorikellojen käyttö on kuitenkin niiden koon takia ongelmallista ympäristöissä, joissa toimii suuri määrä kopioita. [36]

Reaaliaikaiset kellot

Reaaliaikaisiin kelloihin perustuvia aikaleimoja voidaan käyttää tapahtumajärjestyksen määrittämiseen. Tämä menetelmä toimii sitä paremmin, mitä tarkemmin eri laitteiden kellot on tahdistettu. Reaaliaikaleimojen käyttö ei takaa suoritusjärjestyksen vastaavan päivitysten todellista luomisjärjestystä, jos eri laitteiden kellonajat eroavat. Toisaalta se kuitenkin mahdollistaa tietokantojen konvergoitumisen. Lisäksi päivityksiä voidaan levittää laitteilta, jotka eivät ole aikaisemmin kommunikoineet replikointiverkossa ja täten saaneet esimerkiksi loogiseen kelloonsa arvoa. [36]

Ristiriitojen havaitseminen

Optimistisissa replikointijärjestelmissä päivitysten ristiriidat ovat mahdollisia. Ristiriita on tilanne, jossa operaation vaatimat esiehdot eivät täyty. Joissakin järjestelmissä tällaisia tilanteita ei pyritä ratkaisemaan lainkaan. Silloin eri kopiot voivat ajautua erilaisiin tiloihin ja saatavilla oleva data on virheellistä. [36]

Ristiriitojen havaitseminen liittyy läheisesti operaatioiden ajastukseen. Molemmat käyttävät samoja periaatteita ja pyrkivät yhdenmukaistamaan eri kopiot. Ajastuksen tavoin ristiriitojen havaitsemistakin voidaan suorittaa kahdella tavalla: syntaktisesti tai semanttisesti. Syntaktinen havaitseminen perustuu tapahtumajärjestykselle. Semanttisessa havaitsemisessa puolestaan käytetään hyväksi syvempää ymmärrystä muutosoperaatioista. Semanttisilla säännöillä on huomattavasti suurempi ilmaisukyky, mutta niiden tekeminen voi vastaavasti olla monimutkaisempaa. [36]

4.3 Replikointi ja synkronointi haasteellisessa verkossa

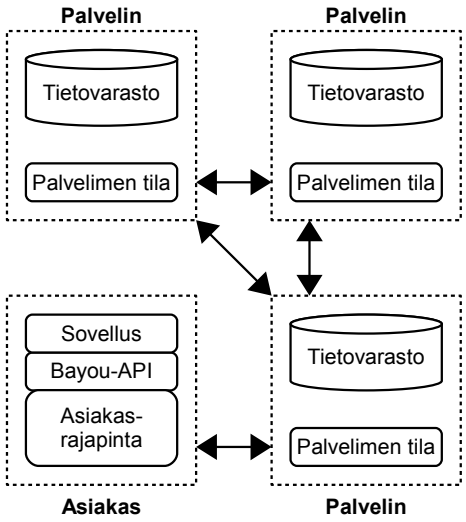
Eric Brewerin [13] mukaan verkossa ei ole mahdollista saavuttaa seuraavia kolmea taetta: yhdenmukaisuus, saatavuus ja jakautumisen sietokyky. Näistä takeista voidaan toteuttaa mitkä tahansa kaksi. Yhdenmukaisuudella tarkoitetaan tilannetta, jossa hajautettu järjestelmä suorittaa operaatioita samalla tavalla kuin niitä suoritettaisiin yhdellä laitteella. Saatavuus edellyttää, että jokainen toimiva verkkolaite pystyy vastaamaan sille lähetettyihin pyyntöihin. Jakautuminen viittaa haasteellisissa verkoissa tyypilliseen tilanteeseen, jossa verkkoon muodostuu saarekkeita. Saarekkeet eivät pysty kommunikoimaan keskenään. Jakautumisen sietokyky sallii saarekkeiden välillä lähetettävien viestien häviämisen. [13]

Toimivan tiedonjakelujärjestelmän toteuttaminen haasteellisessa verkossa vaatii Brewerin esittämien rajoitteiden huomioonottamista. Muun muassa Bayou [38] ja TierStore [8] ovat tällaisia järjestelmiä.

4.3.1 Ratkaisumalli 1: Bayou

Bayou on replikoiva tiedonvarastointijärjestelmä. Se on suunniteltu langattomiin verkkoihin, joissa verkkoyhteydet ovat epäluotettavia. Bayou käyttää monen pääkopion replikointia ja pyrkii lopulta saavutettavaan konsistenssiin. [38]

Bayou on suunniteltu toimimaan verkoissa, joissa laitteet ovat yhteydessä toisiinsa satunnaisesti ja vain pari kerrallaan. Järjestelmän malli ottaa huomioon langattoman verkon ominaispiirteet kuten rajatun yhteydellisyyden, arvokkaan tiedonsiirtokapasiteetin ja yhteyksien katkokset. Bayou on siis tarkoitettu toimimaan myös haasteellisten verkkojen kaltaisissa ympäristöissä, joissa saarekkeiden muodostuminen on mahdollista. Bayoun verkkomalli on esitetty kuvassa 4.13. [38]



Kuva 4.13: Bayoun verkkomalli. [38]

Bayoun ei ole kuitenkaan tarkoitus tarjota näkymätöntä replikointialustaa olemassa oleville tiedosto- ja tietokantasovelluksille. Malli perustuu sille ajatukselle, että sovellusten on oltava tietoisia alla olevan järjestelmän heikosta yhdenmukaisuudesta sekä kirjoitusoperaatioiden mahdollisista ristiriidoista. Tämän lisäksi sovellusten on kyettävä itse ratkaisemaan ristiriidat, sillä ratkaisumekanismi riippuu

kunkin sovelluksen tarpeista. Bayou tarjoaa sovelluksille tavan määrittää, miten ristiriidat havaitaan ja ratkaistaan. Replikointijärjestelmä toimii tämän jälkeen automaattisesti näiden sääntöjen mukaisesti. [38]

Heikon yhteydellisyyden seurauksena Bayou tukee ainoastaan heikosti yhdenmukaista datareplikointia. Eri kopiot eivät siis ole välttämättä aina samassa tilassa. Tämä parantaa huomattavasti datan kirjoitettavuutta, koska muutoksia tehdessä ei tarvitse odottaa kirjoituslukkojen asettamista. Järjestelmä tukee sekä luku- että kirjoitusoperaatioita. Bayoussa dataa voidaan muokata millä tahansa kopiolla. Kukin laite saa lopulta kaikki päivitykset parienvälisten tiedonvaihtojen kautta. Parit vaihtavat keskenään kirjoitusoperaatioita ja sopivat, missä järjestyksessä ne suoritetaan. Tämä replikointimalli on edullinen skaalautuvuuden ja yksinkertaisuuden vuoksi. [38]

Replikointijärjestelmä takaa kaikkien palvelimien päätyvän lopulta yhdenmukaiseen tilaan, mikäli palvelimet saavat samat päivitykset. Tämä perustuu siihen, että päivitykset ajetaan kaikkialla samassa järjestyksessä ja ristiriidat ratkaistaan deterministisesti samalla logiikalla. [38]

Kirjoitusoperaatioilla on Bayoussa kaksi eri tilaa: alustava ja sitoutettu. Asiakkaan lähettämä muutos on ensin alustava. Päivityksen vastaanottava palvelin asettaa muutokselle aikaleiman ja antaa sille oman palvelintunnuksensa. Aikaleimaa ja tunnusta käytetään muutosten järjestyksen määrittämiseen, mutta eri palvelinten kellojen ei kuitenkaan tarvitse olla tahdistettuja. Kellojen tahdistaminen mobiiliverkossa olisi varsin hankalaa. Laitteiden on kuitenkin syytä olla mahdollisimman lähellä samaa aikaa, jotta kirjoitusoperaatiot suoritettaisiin käyttäjän havaitsemassa järjestyksessä. Datapäivitys siirretään sitoutettuun tilaan, kun on tiedossa, että kaikki sitä edeltävät päivitykset on saatu. Bayoussa kullakin replikoitavalla datajoukolla on pääpalvelin, joka tekee sitouttamispäätökset. Jos pääpalvelin ei ole saavutettavissa, toimii järjestelmä käyttäen alustavaa dataa. [38]

Bayou ei pysty takaamaan, että päivitykset sitoututetaan aikaleimojen määräämässä järjestyksessä. Yhdeltä palvelimelta saadut päivitykset suoritetaan aikaleimojen mukaisesti, mutta eri palvelinten tekemät muutokset voivat sitoutua aikaleimojen vastaisesti. Tällainen tilanne voi tulla vastaan, mikäli päivityksiä tekevä palvelin on erillään muista palvelimista sekä sitouttavasta pääpalvelimesta. [38]

Muutosoperaatioiden tallentamiseen Bayou käyttää kirjotuslokia. Alustavat päivitykset säilytetään kumoamislokissa, jotta ne voidaan ajaa uudestaan, mikäli päivityksiä saapuu palvelimelle väärässä järjestyksessä. [38]

4.3.2 Ratkaisumalli 2: TierStore

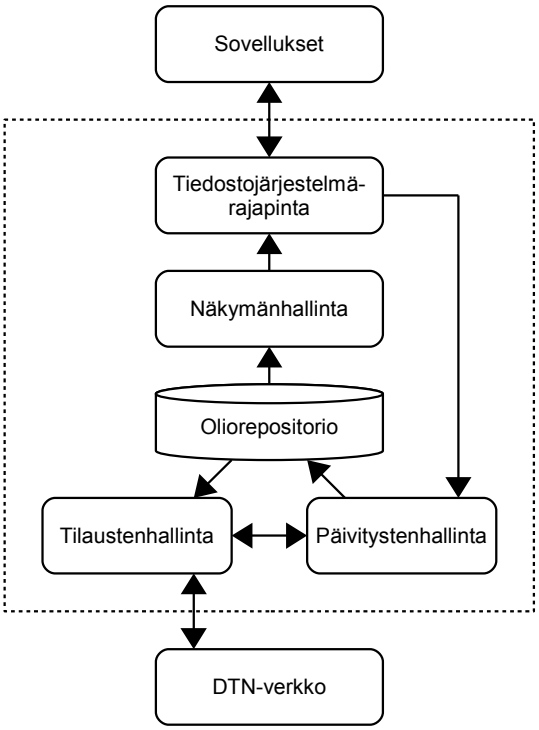
TierStore on hajautettu tiedostojärjestelmä, joka on suunniteltu haasteellisiin verkkoihin. Sen tavoitteena on tarjota haasteellisten verkkojen sovelluksille helppokäyttöinen tiedostorajapinta. TierStore on suunniteltu toimimaan katkeilevien, suuriviiveisten ja matalakaistaisten yhteyksien kanssa. Replikointiin se käyttää tilausjärjestelmään pohjautuvaa monilähetysmekanismia. [8]

TierStore tarjoaa tavallisen rajapinnan tiedostojen käsittelyyn. Jokaisesta muutoksesta tehdään kuitenkin päivitysviesti, joka lähetetään verkon muille noodeille. Jotta järjestelmä toimisi sujuvasti, ei eri tietovarastojen yhdenmukaisuutta taata. Sen sijaan TierStore panostaa tiedon saatavuuteen. [8]

TierStoressa kirjoitusristiriidoiksi määritellään tilanteet, joissa samaa tiedostoa muokataan samanaikaisesti. Ristiriitojen ratkaisu jätetetään ulkopuoliselle taholle. Käytännössä tämä tarkoittaa sovelluksen tekemää automaattista ratkaisua tai käyttäjän manuaalista päätöstä. [8]

Kaistankäytön rajoittamiseksi TierStore mahdollistaa julkaisujen luomisen. Käytännössä julkaisulla tarkoitetaan tiedostojärjestelmän alipuuta. Verkkolaitteet voivat tämän jälkeen tilata sitä kiinnostavat julkaisut luomalla tilaustiedon ennalta määriteltyyn hakemistoon. Lisäksi noodit järjestetään levityspuuhun, jota käytetään monilähetyksen mahdollistamiseksi. Tällä mekanismilla vältetään tarpeetonta tiedonsiirtoa. [8]

TierStore koostuu eri komponenteista, joilla kullakin on oma tehtävänsä. Komponentit näkyvät kuvassa 4.14. Tiedostojärjestelmärajapinta tarjoaa sovelluksille keinot muokata dataa. Näkymänhallintaosa toteuttaa lukuoperaatiot ja ratkaisee päivitysten väliset ristiriidat, jotta näkymä tietoon on yhdenmukainen. Oliorepositoriossa säilytetään varsinainen data. Kirjoitusoperaatiot kulkevat päivitystenhallinnan kautta ja muun verkon kanssa viestitään tilaustenhallintakomponentin välityksellä. [8]



Kuva 4.14: TierStore-järjestelmän rakenne yhdellä verkkonoodilla. [8]

Muutosten seurantaan TierStoressa käytetään versiovektoreita. Kullakin noodilla on oma laskuri, jota kasvatetaan aina, kun tehdään muutos. Eri noodeilta kerättyjen laskureiden arvot määrittävät versiovektorin, joka puolestaan kertoo päivitysten järjestyksen. TierStore on suunnattu suhteellisen pieniin ympäristöihin, joissa noodeja on enimmillään joitakin satoja. Tästä syystä versiovektoreiden ei oleteta kasvavan liian suuriksi. [8]

Suuriviiveisten verkkojen takia päivitysten levittämisessä pyritään välttämään tarpeettomien viestien lähettämistä. TierStore levittää päivityksiä verkkoon sitä mukaa, kun niitä tehdään. Järjestelmä kuitenkin edellyttää alemmilta kerroksilta, että kaikki viestit välitetään lopulta vastaanottajille. [8]

Luku 5

Toteutusmalli

Tässä luvussa esitetään ehdotus haasteellisessa verkossa toimivan hajautetun osoitepalvelun rakenteelle. Aluksi luvussa esitellään toteutusympäristö eli verkko, johon osoitepalvelu on kohdistettu. Haasteellisen verkon ominaisuudet ja laitteiden resurssit rajoittavat osoitepalvelun toimintamahdollisuuksia. Toteutusympäristön pohjalta voidaan palvelulle tehdä teknologioita ja tietoliikennettä koskevat valinnat. Nämä valinnat käsittelevät muun muassa hajautettujen tietokantojen yhdenmukaisuutta, asiakas- ja replikointirajapintoja, levitysmekanismeja sekä pääsynhallintaa.

5.1 Toteutusympäristö

Tässä osiossa kuvataan kohdeverkon ominaisuuksia sekä verkossa käytettäviä palvelin- ja asiakaslaitteita. Osoitepalvelu on tarkoitus toteuttaa haasteelliseen verkkoon, mikä asettaa verkkorajapinnoille tiettyjä vaatimuksia. Luotettavissa verkoissa toimivat ratkaisut eivät ole välttämättä käyttökelpoisia haasteellisessa verkossa sen rajoitteiden ja ominaispiirteiden vuoksi. Kohdeverkon oletetaan kärsivän erilaisista haasteista muun muassa siirtokapasiteetin, yhteydellisyyden sekä viiveiden suhteen. Lisäksi verkossa toimivat palvelimet ja etenkin asiakaslaitteet vaativat resurssiensa vuoksi erityishuomiota järjestelmää suunniteltaessa.

Osoitekirjan alustava sisältö luodaan keskitetysti etukäteen. Tämä käsittää sekä organisaatiohierarkian että olioiden attribuutit. Eri palvelimet kuitenkin alustetaan datan eri alijoukoilla. Kun verkko hajautetaan, on eri palvelimilla oma osionsa alustavasta tiedosta. Näitä osoitetietoja voidaan hajautuksen jälkeen päivittää. Osoitedatan hajauttamisella pyritään muun muassa vähentämään verkkoliikennettä ja tilankäyttöä laitteilla. Hajautus vähentää myös mahdollisen tietovuodon

vaikutuksia, sillä yhden palvelimen haltuunotto ei paljasta koko osoitepalvelun sisältöä.

Koska palvelimet alustetaan keskitetysti, voidaan myös replikoinnin ja synkronoinnin topologiaa suunnitella ennen hajautusta. Vaikka verkko onkin haasteellinen, eivät osoitepalvelimet ole kovinkaan liikkuvia, joten topologia ei muutu usein.

Laitealustat ja -resurssit

Osoitetiedon jakeluun käytettävät laitteet ovat pääasiassa palvelimia, joilla ajetaan Linux-käyttöjärjestelmää. Jakeluina käytetään Ubuntun ja Debianin eri versioita, mutta myös muut jakelut ovat mahdollisia. On myös mahdollista, että osoitekirjapalvelua halutaan ajaa Windows-laitteilla. Käytettävät ohjelmistot ja alustat on siis valittava siten, että järjestelmä voidaan toteuttaa halutuille alustoille.

Palvelinlaitteilla oletetaan olevan riittävästi laskentatehoa ja käyttömuistia hakemisto-ohjelmiston sujuvaan ajamiseen. Käytettävissä olevan massamuistin oletetaan kullakin laitteella riittävän laitteen oman hakemisto-osion alkutilan varmuuskopion tallentamiseen. Tätä alkutilaa käytetään hyväksi replikointi- ja synkronointijärjestelmien takaisinpaluussa. Lisäksi kunkin laitteen tulee tallentaa sekä itse luodut että muualta saapuneet hakemistopäivitykset pitkäaikaista säilytystä varten. Näin toimitaan siksi, että päivityksiä voidaan joutua suorittamaan tai lähettämään jälkikäteen uudestaan. Tämä on edellytys synkronointijärjestelmän toimimiselle.

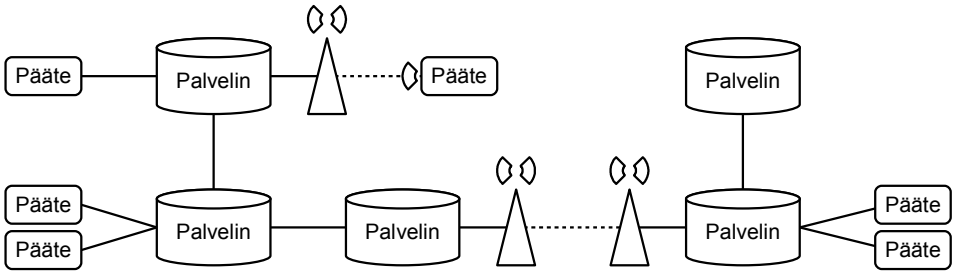
Osoitepalvelua on tarkoitus pystyä käyttämään useilla erilaisilla päätelaitteilla. Kaikilla alustoilla ei ole mahdollista ajaa vaativampia asiakassovelluksia, vaan palvelun hyödyntäminen asiakkailla tapahtuu useimmiten sanomarajapinnan kautta. Joissakin käytössä olevista laitteista on hyvin rajattu merkistö, mikä täytyy ottaa huomioon palvelua suunniteltaessa.

Laitteiden toimintaikä

Verkon on tarkoitus toimia haasteellisessa ympäristössä, jossa voi esiintyä myös aktiivisia vastatoimia. Täten yhteyksien ja laitteiden toiminnasta ei voida antaa takeita, ja toimintaiät voivatkin vaihdella suuresti. Palvelimet saattavat rikkoutua pysyvästi ilman varoitusta ja hävitä verkosta.

Verkkorakenne

Osoitepalvelun pohjana käytetään DTN-periaatteella (Delay-Tolerant Networking) toimivaa sanomaverkkoa. Tämän lisäksi asiakkaat voivat yhdistää palvelimelle IP-verkosta. Sanomaverkossa ajettava osoitepalvelu toimii sekä asiakas-palvelin -mallilla sekä vertaisverkkona. Loppukäyttäjien laitteet ovat palvelussa asiakkaita ja hakemistolaitteet palvelimia. Replikointi- ja synkronointijärjestelmät sen sijaan toimivat vertaisverkon periaatteella. Osoitepalvelimet ovat tasa-arvoisia ja välittävät päivitystietoa toisilleen. Kuvassa 5.1 on esitettu hahmotelma sanomaverkon topologiasta. Laitteet voivat olla kytköksissä toisiinsa langallisesti tai langattomasti. Kukin päätelaite on rekisteröitynyt kerrallaan yhdelle palvelimelle.



Kuva 5.1: Esimerkki pienen sanomaverkon topologiasta, jossa on käytössä langattomia sekä langallisia yhteyksiä.

Siirtokapasiteetti, yhteyksien luotettavuus ja jonotusajat

Käytössä olevan verkon siirtokapasiteetti ja -viiveet voivat vaihdella suuresti eri linkkien välillä. Verkko koostuu useista eri teknologioista, ja linkitkin voivat olla langattomia tai langallisia. Vähäisen kaistanleveyden vuoksi osoitepalvelun viestintä on pyrittävä pitämään alhaisena niin viestien koossa kuin lukumäärässäkin.

Kohdeverkon yhteyksien oletetaan olevan hyvin epäluotettavia. Laitteet voivat hävitä verkosta tai vaihtaa paikkaansa ilmoittamatta. Yhteydet voivat kärsiä erilaisista verkko-ongelmista kuten väliaikaisista tai pysyvistä katkoista. Noodit kuitenkin huomaavat omien linkkiensä katkeamisen ja palaamisen, joten ongelmiin on mahdollista reagoida ohjelmistopuolella.

Kohdeverkossa liikkuu osoitetietojen lisäksi myös merkittävä määrä muiden palveluiden luomaa liikennettä. Osa liikenteestä on hyvinkin aikakriittistä, minkä johdosta sen prioriteetti välityslaitteissa voi olla osoitedataa korkeampi. Tämän seurauksena osoitepäivitysten välittämisessä saattaa esiintyä viiveitä.

Verkkorajapinnat

Osoitepalvelua on tarkoitus pystyä käyttämään kahden eri verkon välityksellä. Nämä ovat yhteydetön sanomaverkko ja yhteydellinen TCP/IP-verkko. Sanomaverkkoa käytetään laajemmassa mittakaavassa. TCP/IP-verkkoa voidaan hyödyntää silloin, kun sellainen on saatavilla. Sanomaverkko on viestipohjainen järjestelmä, jossa sovellukset kommunikoivat käyttäen IMAP- (Internet Message Access Protocol) ja SMTP-sähköpostirajapintoja (Simple Mail Transfer Protocol). Näitä rajapintoja voidaan käyttää palvelinten väliseen viestintään. Sanomaverkon päätelaitteet eivät kuitenkaan tukeudu sähköpostiin, vaan viestivät palvelimille käyttäen omaa asiakasviestiformaattiaan. TCP/IP-verkolla voidaan tarkoittaa laajempaa WAN-verkkoa (Wide Area Network) tai tavallista lähiverkkoa, jossa asiakaslaitteet yhdistävät paikalliselle osoitepalvelimelle. Osoitepalvelussa voidaan siis osittain hyödyntää TCP/IP-yhteyksiä, mutta toimivuus yhteydettömässä sanomaverkossa on myös taattava.

Tietokantojen konvergoituminen

Osoitepalvelussa tietojen yhdenmukaistumiselle asetetaan suuri prioriteetti. Eri palvelimet eivät saa ajautua pysyvästi eri tiloihin, jotta palvelun pitkäaikainen käyttö olisi mahdollista. Vahvaa yhdenmukaisuutta ei tässä ympäristössä kuitenkaan voida saavuttaa. Tästä johtuen tavoitteena on pyrkiä lopulta saavutettavaan konsistenssiin. Toisin sanoen eri hakemistokopioiden tulisi ajan kuluessa konvergoitua samanlaiseen tilaan. Monen pääkopion järjestelmässä tämä tarkoittaa käytännössä sitä, että ristiriitojen havaitseminen ja ratkaiseminen on tärkeää. Tilapäisiä epäyhdenmukaisuuksia järjestelmässä saa ilmetä, mutta konsistenssiin on pyrittävä niin hyvin kuin mahdollista.

Tietoturva

Sanomaverkossa viestivien käyttäjien tunnistaminen tehdään alemmilla verkkokerroksilla. Osoitepalvelun on kuitenkin erotettava käyttäjät toisistaan, jotta pääsynhallinta toimisi. IP-verkkojen oletetaan toteutusympäristössä olevan turvallisia, joten ne eivät tarvitse sovelluskerroksella liikenteen suojausta. Käyttäjät on kuitenkin edelleen tunnistettava.

Osoitekirjan data on hajautettava palvelimille siten, ettei kaikkea tietoa ole saatavilla yhdestä paikasta. Lisäksi käyttäjillä on oltava pääsy ainoastaan rajattuun määrään dataa. Näiden vaatimusten tarkoitus on pienentää mahdollisten tietovuotojen laajuutta.

5.2 Teknologiavalinnat

Tässä osiossa tehdään järjestelmän kannalta keskeiset teknologiavalinnat. Valinnat koskevat palvelun tarvitsemaa tietovarastoa sekä käytettyjä tiedonsiirtoformaatteja.

5.2.1 Tietomalli ja hakemisto

Osoitepalveluun on valittava tietomalli, joka soveltuu hyvin organisaatiorakenteen sekä osoitetietojen tallentamiseen. Lisäksi valitun toteutuksen hyödyntämän teknologian olisi suotuisaa olla yleisesti käytetty, jotta integrointi asiakassovelluksiin helpottuisi. Varteenotettavimmat vaihtoehdot tiedon varastointiin ovat relaatiomalli sekä hierarkkinen malli. Relaatiotietokannat ovat hyvin laajassa käytössä, ja mallia käyttäviä tietokantaohjelmistoja löytyy runsaasti [15]. Hierarkkinen malli puolestaan soveltuu erinomaisesti organisaation kuvaamiseen, joten hakemistopohjainen ratkaisu olisi suotuisa tiedon jäsentelyn puolesta. Verkkomallin, oliomallin tai moniulotteisen mallin käytöstä ei puolestaan saada merkittäviä etuja hierarkkiseen malliin nähden.

Olemassa olevista teknologioista LDAP-hakemistopalvelimet täyttävät vaaditut ehdot parhaiten. LDAP-protokolla sekä -palvelimet pohjautuvat hierarkiaan ja mahdollistavat monipuolisen tiedonhaun ja datan muokkaamisen. Protokolla on myös käytössä monissa viestintäsovelluksissa. Näistä syistä osoitepalvelu toteutetaan hyödyntämällä LDAP-ohjelmistoja.

Palvelinohjelmistoa valittaessa kriteereiksi asetettiin lähdekoodin avoimuus sekä alustatuki vähintäänkin Linuxille. Windows-käyttöjärjestelmien tukeminen laskettiin eduksi. Varteenotettavimmat vaihtoehdot olivat siis 389 Directory server, ApacheDS, OpenDJ sekä OpenLDAP. Näistä OpenLDAP on pitkäikäisin ja melko hyvin dokumentoitu. Lisäksi se on yleisesti käytetty ja siten laajalti testattu. Palvelin on myös käännettävissä useille eri aluistoille. OpenLDAPin muihin etuihin lukeutuvat muun muassa monipuoliset pääsynhallintasäännöt, laajennettavuus useilla valmiilla päällysosilla sekä erilaiset vaihtoehdot taustaosaksi. [23]

OpenLDAPiin on saatavilla useita erilaisia taustaosia. Taustaosan tehtävä on hoitaa tiedon varastointi ja hakeminen. Tiedon tallentamisen lisäksi on olemassa joitakin erikoistarkoituksiin kohdistettuja taustaosia. Näihin tarkoituksiin lukeutuvat muun muassa välityspalvelin, hakemiston valvonta sekä erillisen skriptin suorittaminen. [23]

Yleisimmät ratkaisut varsinaiseen tiedon säilyttämiseen OpenLDAP-ympäristössä ovat Oracle Berkeley Databaseen pohjautuvat BDB- ja HDB-taustaosat (Berkeley

Database, Hierarchical Database). Ne ovat hyvin samankaltaisia, mutta HDB tukee alipuiden uudelleennimeämistä eli siirtoa. MDB (Memory-Mapped Database) on uudempi, Berkeley Database -ratkaisujen korvaajaksi kehitetty taustaosa. Se toteuttaa samat ominaisuudet kuin HDB ja toimii vähäisellä konfiguroinnilla. [23]

OpenLDAP-hakemiston taustalla on mahdollista käyttää myös SQL-relaatiotietokantaa. Tätä varten on olemassa NDB- ja SQL-taustaosat. NDB:llä hakemiston data voidaan tallentaa MySQL-tietokantaan. Tällöin hierarkia tallennetaan yhteen tauluun ja itse oliot hajaututetusti useisiin eri tauluihin. NDB:n tarkoitus on toimia yleiskäyttöisenä taustaosana. Se on kuitenkin kokeellinen, joten siitä puuttuu monia LDAP-toiminnallisuuksia. [24]

SQL-taustaosa mahdollistaa relaatiotietokannassa säilytetyn tiedon esittämisen LDAP-hakemistona. Sen ei siis ole tarkoitus toimia yleiskäyttöisenä taustaosana. SQL-taustaosa voidaan konfiguroimalla yhdistää mihin tahansa SQL-skeemaan. Tällöin LDAP-kyselyt muutetaan SQL-kyselyiksi metatietoa käyttäen. [23]

Koska SQL-tuki ei tuo palveluun merkittävää lisäarvoa, soveltuvat taustaosiksi parhaiten HDB ja MDB. Organisaatiorakenteen tallentaminen LDAP-hakemistopuuhun on melko suoraviivaista. Hierarkkisen organisaation kuvaamiseen riittää yksinkertainen skeema, joka määrittelee osastoja ja henkilöitä varten luokat. Itse osoitteet voidaan tallentaa attribuuteiksi. Palvelun on tarkoitus tukea myös LDAPin tarjoamia moniarvoisia attribuutteja. Niiden avulla olioihin voidaan tallentaa esimerkiksi useampi puhelinnumero tai sähköpostiosoite. Palvelun rajapintojen on kuitenkin tuettava moniarvoisten attribuuttien käsittelyä tarjoamalla siihen soveltuvat operaatiot.

Tässä toteutuksessa kullekin palvelimelle syötetään alustavat osoitetiedot ennen järjestelmän hajautusta. Käytännössä tämä ei kuitenkaan ole edellytys palvelun toiminnalle, vaan tietojen yhdenmukaistaminen voidaan tehdä myös synkronoimalla palvelimet hajautetussa verkossa.

5.2.2 Päivitysformaatti

Asiakasrajapinta

Asiakasrajapinnalle on valittava asetetut vaatimukset täyttävä tiedonsiirtoformaatti. Kohdeverkon ominaispiirteet asettavat rajoituksia tiedon välittämisessä käytetylle tietomuodolle. Formaatin on siis tuettava tarvittavia hakemisto-operaatioita, sovelluttava haasteelliseen verkkoon sekä mukauduttava asiakasviestiformaattiin ja sen merkistöön.

LDIF-formaatilla voidaan kuvata LDAPin muutosoperaatioita, ja DSML mahdollistaa niiden lisäksi myös hakujen ilmaisemisen. Kyseiset formaatit eivät kuitenkaan välttämättä sovellu hyvin haasteelliseen verkkoon.

Koska siirtokapasiteetti verkossa saattaa olla vähäistä, olisi hakemistotiedon siirtämisessä käytetyn formaatin oltava mahdollisimman kompakti. Operaatioiden DSML-kuvauksessa on merkittävästi redundanttia dataa, joten sen käyttö ei ole suotavaa. LDIF on huomattavasti tiiviimpi muoto, mutta tätä toteutusta koskevien ominaisuuksien perusteella on siitäkin mahdollista jättää tietoa pois. Tarjottavien operaatioiden ei tarvitse myöskään olla yhtä monipuolisia kuin LDAP-rajapinnan. Esimerkiksi hakuoperaation suodintukea voidaan yksinkertaistaa. Rajapinta muuttuu tällöin helppokäyttöisemmäksi, mutta tärkeimmät hakutoiminnallisuudet pystytään säilyttämään.

Asiakasviestiformaatin käyttämä merkistö on hyvin rajoitettu, mikä täytyy ottaa huomioon tiedonsiirtoformaatissa. Esimerkiksi täyden LDIF- tai DSML-tuen tekeminen ei ole mahdollista ilman merkistömuutoksia. Merkkejä täytyisi joko jättää pois tai muuttaa muiksi merkeiksi. Mikäli kaikki tieto halutaan säilyttää, on käyttöön otettava koodaus, jossa useampi sanomamerkki vastaa yhtä LDIF- tai DSML-merkkiä.

Osoitepalveluviestien on mukauduttava käytettyyn asiakasviestiformaattiin. Viestit ovat tekstimuotoisia, ja niiden sisältöä ryhmitellään rivinvaihtoja ja kauttaviivoja erottimina käyttäen. Käytännössä osoitepalvelun viestit voidaan siirtää vapaamuotoisena tekstinä asiakasviestin sisällä tai operaatioiden parametrit voidaan ryhmitellä asiakasformaatin erottimilla. Ensin mainittu menetelmä mahdollistaisi esimerkiksi LDIF-datan siirtämisen asiakasviestissä, mikäli merkistön epäyhteensopivuutta ei huomioida. Jälkimmäinen vaihtoehto on parempi järjestelmien yhteentoimivuuden puolesta, koska silloin osoitepalveluviestien parsiminen voidaan tehdä samalla menetelmällä kuin muidenkin asiakasviestien. Asiakasviestiformaattia siis laajennetaan osoitepalveluun soveltuvalla viestirakenteella. Suunniteltu viestirakenne esitellään kohdassa 5.4.2.

Palvelinten väliset rajapinnat

Osoitepalvelimet viestivät keskenään sanomaverkossa. Ne eivät kuitenkaan kärsi samoista rajoitteista kuin jotkin päätelaitteet. Tästä johtuen palvelinten välisessä viestinnässä ja asiakasrajapinnalla on mahdollista käyttää eri päivitysformaatteja. Asiakasrajapinta vaatii ominaispiirteidensä johdosta oman viestimuotonsa, mutta palvelinten välillä voidaan käyttää ilmaisukykyisempääkin formaattia.

LDIF-muodon käyttämisestä replikoinnissa ja synkronoinnissa saadaan joitakin etuja. LDIF-muotoiset päivitykset voidaan lukea suoraan LDAPia tukeviin järjes-

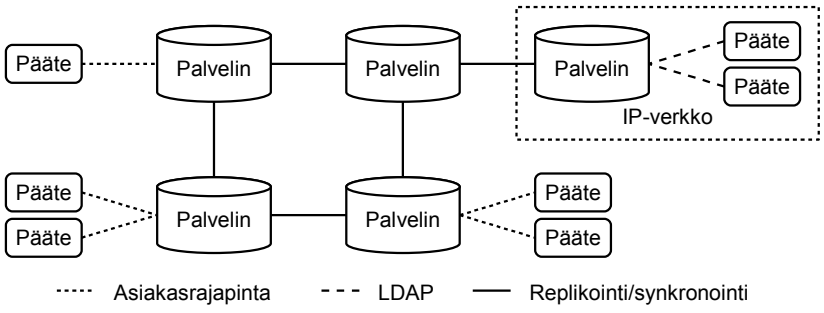
telmiin. Näin alun perin asiakasrajapinnan kautta tehtyjä päivityksiä ei tarvitse muuttaa LDIF-muotoon erikseen joka palvelimella. Lisäksi formaatti mahdollistaa asiakasrajapinnan formaattia monipuolisempien päivitysten tekemisen, mikäli sellaisia verkossa halutaan levittää. SMTP:llä välitettävät replikointi- ja synkronointiviestit voivat puolestaan noudattaa mitä tahansa järjestelmään soveltuvaa siirtoformaattia, jonka sisälle LDIF-muotoista dataa voidaan tallentaa.

5.3 Palvelun rakenne

Palvelun rakenne on kuvattu tässä osiossa. Kuvaus kattaa sekä osoitepalveluverkon rakenteen että yksittäisen palvelimen sisäisen rakenteen komponentteineen ja rajapintoineen.

Verkkorakenne

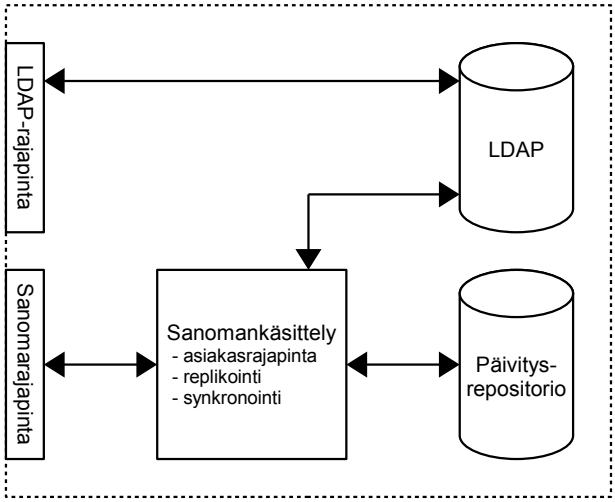
Yksinkertaistettu kuva palvelun rakenteesta on esitetty kuvassa 5.2. Palvelu rakentuu palvelinlaitteille, joilta erilaiset päätelaitteet voivat kysellä osoitetietoja. Osoitekyselyitä ja -päivityksiä voidaan suorittaa sanomaverkon välityksellä asiakasformaatissa. Vaihtoehtoisesti asiakas voi tehdä kyselyitä käyttäen LDAPia, mikäli TCP/IP-yhteys on muodostettavissa. Päätelaitteiden ja palvelinten välinen viestintä tapahtuu asiakas-palvelin -mallilla. Palvelimet välittävät toisilleen hakemistopäivityksiä replikointi- ja synkronointijärjestelmän välityksellä. Palvelinten välinen viestintä toimii vertaisverkon tavoin.



Kuva 5.2: Esimerkkikuva osoitepalveluverkosta ja sen eri rajapinnoista.

Palvelinrakenne

Kuvassa 5.3 on esitetty palvelimen sisäinen rakenne sekä viestintärajapinnat. Osoitekirjan dataa säilytetään LDAP-hakemistossa. Kaikki laitteelle saapuneet päivitykset tallennetaan päivitysrepositorioon, mikä mahdollistaa synkronoinnin ja hakemiston takaisinpaluun. Palvelimella ajetaan myös ohjelmaa, joka käsittelee sanomarajapinnan kautta tulleita viestejä. Tämä sanomankäsittelykomponentti toteuttaa kolmen eri palvelun rajapinnat, joista kullekin on oma viestintäformaatti ja viestinprosessointi. Sanomankäsittely koostuu siis asiakas-, replikointi- ja synkronointirajapinnoista. Asiakasrajapinnan tarkoitus on mahdollistaa osoitetietojen haku ja käsittely asiakaslaitteilla ja -sovelluksilla. Täten sanomankäsittely vastailee hakuviesteihin ja prosessoi asiakkailta tulevat päivitykset. Replikointijärjestelmän tehtävä on vastaanottaa ja aktiivisesti välittää palvelimelle saapuvia päivityksiä eteenpäin. Synkronointijärjestelmä puolestaan käsittelee synkronointiviestit, kun palvelinten data halutaan yhdenmukaistaa.



Kuva 5.3: Palvelimen sisäinen rakenne.

Sanomankäsittely siis kommunikoi LDAP-palvelimen kanssa. Tämän lisäksi se kirjoittaa ja lukee päivitysrepositoriota. Saapuneet päivitykset kirjoitetaan repositorioon ja synkronoitavat tiedot luetaan sieltä. Verkko-ongelmien aiheuttamaa virheellistä päivitysjärjestystä korjattaessa palvelimelle suoritetaan takaisinpaluu, jolloin hakemisto alustetaan ja repositoriossa olevat päivitykset ajetaan uudelleen hakemistoon.

Palvelinta voidaan käyttää sekä LDAP- että sanomarajapinnan avulla. LDAP kuitenkin mahdollistaa tässä toteutusmallissa ainoastaan tiedon hakemisen. Koska

LDAP-rajapinnan kautta ei sallita tehtävän muutoksia, ei sen tarvitse olla kytköksissä replikointijärjestelmään.

5.4 Asiakasrajapinta

Asiakasrajapinnan tarkoitus on mahdollistaa osoitepalvelun käyttäminen haasteellisen verkon päätelaitteilla. Kaikki verkon laitteet eivät kykene viestintään LDAP-rajapinnan avulla mutta pystyvät kuitenkin kommunikoimaan sanomilla. Lisäksi kaikkialla verkossa ei ole lainkaan mahdollista muodostaa TCP/IP-yhteyksiä. Näistä syistä osoitepalvelun käyttö on tehtävä sanomaviesteillä.

Asiakasrajapintaa koskee joitakin rajoituksia viestintäformaatin ja merkistön suhteen. Tässä osiossa suunnitellaan formaattikuvaus, joka kattaa kyseiseltä rajapinnalta vaaditut hakemisto-operaatiot.

5.4.1 Hakemisto-operaatiot

Palvelulle asetetuista vaatimuksista johtuen kaikkia hakemisto-operaatioita ei tarvitse toteuttaa kaikille rajapinnoille. Sanomarajapinnan kautta on pystyttävä ainoastaan muokkaamaan ja hakemaan osoitetietoja.

LDAPiin on määritelty 11 eri LDAP-operaatiota. Osa näistä operaatioista koskee asiakkaan ja palvelimen välisen LDAP-yhteyden tilaa. Nämä käsittävät palvelimelle autentikoinnin, yhteyden hylkäämisen ja salauksen käyttöönoton. Osalla operaatioista muokataan hakemiston sisältöä tai suoritetaan haku tai vertailu. Lisäksi LDAPilla on mahdollista pyytää palvelinta perumaan suorituksessa oleva operaatio tai suorittamaan erikseen määritelty laajennettu operaatio. [37]

Koska sanomaverkko on yhteydetön, ei LDAP-yhteyden tilaa koskeville operaatioille tarvitse tehdä vastineita. Näiden lisäksi asiakasrajapinnalla tarpeettomia operaatioita ovat vanhan operaation peruutus, laajennettu operaatio sekä vertailu.

LDAP-olioita voidaan luoda hakemistoon lisäysoperaatiolla [37]. Tässä toteutusmallissa olioita ei ole tarpeen luoda asiakasrajapinnan kautta osoitekirjan hajauttamisen jälkeen, joten operaatiota ei ole tuettava. Sen toteuttaminen olisi kuitenkin mahdollista. Tätä varten lisäyssanoman olisi määritettävä vähintäänkin ylemmän solmun nimi ja lisättävän olion RDN-nimi. Lisäksi olioon olisi mahdollista sisällyttää attribuutteja, ja sille voitaisiin antaa luokkamäärite.

Olioiden poistaminen tapahtuu poisto-operaatiolla [37]. Tässä toteutusmallissa oliota ei tarvitse poistaa, mutta tällekin operaatiolle olisi mahdollista tehdä tu-

ki. Tätä varten poistoviestissä olisi ilmoitettava kohdeolion yksilöivä tunniste. On kuitenkin huomattava, että LDAP tukee ainoastaan lehtisolmujen poistamista. Mikäli rekursiivista poistamista halutaan tukea, on sanomarajapintakomponentin suoritettava ensin haku, jolla selvitetään kaikki alipuun noodit, ja tämän jälkeen poistettava ne lehtisolmuista alkaen.

Alipuiden siirto voidaan LDAPissa tehdä siirto-operaatiolla [37]. Se toimii kuitenkin vain tietyissä OpenLDAPin taustaosissa kuten HDB:ssä ja MDB:ssä [23]. Tässä suunnitelmassa operaatiolle ei tehdä vastinetta. Olioiden siirtäminen aiheuttaa tietyissä tilanteissa ongelmia hajaututetussa ympäristössä. Ongelma syntyy esimerikiksi, jos hakemistoa muokkaavalla palvelimella ei ole siirrettävää oliota. Tällöin olion tiedot täytyisi saada muualta verkosta, jotta se voitaisiin luoda.

LDAPin muokkausoperaatiolla käsitellään olioiden attribuutteja. Se käsittää kolme alioperaatiota, jotka ovat lisäys, poisto ja korvaaminen. Lisäysoperaatiolla voidaan attribuutille lisätä arvo tai arvoja. Poisto mahdollistaa valitun attribuuttiarvon poistamisen sekä koko attribuutin tyhjentämisen. Korvausoperaatiolla attribuutin arvot voidaan korvata uusilla arvoilla. Operaatio kuitenkin poistaa kaikki vanhat arvot, eikä tiettyä arvoa ole mahdollista valikoida korvattavaksi. Tähän rajapintaan suunnitellaan vastineet kaikille kolmelle attribuuttien muokkausoperaatiolle. [37]

LDAPin hakutoiminnallisuus mahdollistaa hyvinkin monimutkaisten hakujen tekemisen [37]. Asiakasrajapinta pyritään kuitenkin pitämään yksinkertaistettuna, joten esimerkiksi erilaisia suotimia ei tarvita. Haulla täytyy kuitenkin tarvittaessa pystyä pyytämään vain määrätyt attribuutit, koko kohdeolio tai kokonainen alipuu.

5.4.2 Tiedonsiirtoformaatti

LDIF-muotoisen datan käsittely asiakasrajapinnalla on tehotonta ja epäkäytännöllistä. Tästä johtuen on käyttöön otettava tiedonsiirtomuoto, joka soveltuu asiakasympäristöön paremmin.

Osoitepalveluviestien formaatti on kuvattu listauksessa 5.1 EBNF-muodossa [17] (Extended Backus-Naur Form). Viesti (`"viesti"`listauksessa) alkaa palvelutunnisteella (`"palvelu"`), jotta data voidaan välittää oikealle sovellukselle. Tätä seuraa pääsyoikeudet määrittävä käyttäjätunniste (`"käyttäjä"`), jota hyödynnetään myös päivitysviestien yksilöimisessä. Käyttäjätunnisteen käyttöoikeus on kuitenkin vahvistettava ainakin ensimmäisellä palvelimella, jotta palvelussa ei voida esiintyä väärillä tunnuksilla.

Tunnistetietojen jälkeen viestissä määritetään itse operaatiot ("`operaatio`") rivinvaihdoilla erotettuina. Yhdessä viestissä voi olla yksi tai useampi operaatio, joista kukin alkaa kohdeolion tunnuksella. Oliotunnusta seuraa operaation nimi, joka kertoo, onko kyse lisäyksestä, poistosta, korvaamisesta vai kyselystä.

Listaus 5.1: Asiakasrajapinnan muutos- ja kyselyviestiformaatti EBNF-muodossa.

```
viesti   = palvelu, käyttäjä, { operaatio, rivinvaihto }- ;
palvelu  = "Osoitepalvelu/pyyntö", rivinvaihto;
käyttäjä = tekstijono, rivinvaihto;

operaatio = ( lisäys | poisto | korvaus | kysely );

lisäys  = kohde, "/Lisäys", { "/", attribuutti, ":", arvo }- ;
poisto  = kohde, "/Poisto", { "/", attribuutti, [ ":", arvo ] }- ;
korvaus = kohde, "/Korvaus", { "/", attribuutti, ":", arvo }- ;
kysely  = kohde, "/Kysely", "/" | "/*" | { "/", attribuutti }- ;

kohde       = tekstijono;
attribuutti = tekstijono;
arvo        = tekstijono;

rivinvaihto = ? ASCII-merkki 13 ?, ? ASCII-merkki 10 ?;
tekstijono  = { ? kaikki näkyvät merkit ? }- ;
```

Lisäysoperaatiolla attribuuteille voidaan lisätä arvoja. Tällöin viestissä määritellään attribuutin nimi ("`attribuutti`") ja uusi arvo ("`arvo`"). Nimi-arvo -pareja voi olla useampi, tällöin ne erotellaan kauttaviivalla.

Poisto-operaatiossa voidaan määritellä yksi tai useampi tyhjennettävä attribuutti. Määrittely tapahtuu antamalla attribuuttien nimet ("`attribuutti`") kauttaviivalla eroteltuina. Moniarvoisista attribuuteista voidaan haluttu arvo ("`arvo`") poistaa antamalla se attribuuttinimen ja kaksoispisteen jälkeen.

Korvausoperaatio poistaa kaikki halutun attribuutin ("`attribuutti`") arvot ja korvaa ne annetulla arvolla ("`arvo`").

Kyselyoperaatioon sisällytetään hakuparametrit, jotka voidaan määritellä kolmella eri tavalla. Tyhjää kenttää käytetään noutamaan kaikki kohteen attribuutit. Asteriskia käytetään pyytämään kohdeolion kaikkien lasten kaikki attribuutit. Viimeinen käyttötapa on haettavien attribuuttien määrittely erikseen. Haettavat attribuuttinimet ("`attribuutti`") erotetaan kauttaviivalla.

Muutosviesteihin ei asiakasrajapinnalla lähetetä vastausta. Asiakas ei siis tässä mallissa saa kuittausta operaation onnistumisesta. Kyselyviesteihin luonnollisesti vastataan osoitetiedoilla. Vastausviestin formaatti on kuvattu listauksessa 5.2.

Vastaus ("`viesti`"listauksessa) koostuu palvelutietojen ("`palvelu`") lisäksi kyseltyjen olioiden tunnuksista ("`kohde`") ja attribuuttidatasta ("`attribuutti`"ja "`arvo`").

Listaus 5.2: Asiakasrajapinnan vastausviestiformaatti EBNF-muodossa.

```
viesti     = palvelu, { oliotiedot, rivinvaihto }- ;
palvelu    = "Osoitepalvelu/vastaus", rivinvaihto;

oliotiedot  = kohde, { "/", attribuutti, ":", arvo }- ;

kohde       = tekstijono;
attribuutti = tekstijono;
arvo        = tekstijono;

rivinvaihto = ? ASCII-merkki 13 ?, ? ASCII-merkki 10 ?;
tekstijono  = { ? kaikki näkyvät merkit ? }- ;
```

Asiakasrajapinnan kautta tulevat päivitykset muutetaan ensimmäisellä palvelimella LDIF-muotoon. Lisäksi päivitykselle annetaan yksilöllinen tunnus muun muassa lähettäjän, palvelimen ja ajanhetken perusteella. Tämän jälkeen päivitys voidaan tallentaa paikalliseen repositorioon sekä suorittaa paikallisesti ja lähettää replikointijärjestelmän kautta muille palvelimille. Replikointiviesteinä saapuvat päivitykset tallennetaan ja suoritetaan myös muilla palvelimilla.

Asiakasviestejä LDIF-muotoon muutettaessa on päivitykset toisinaan jaettava useampaan tiedostoon. Tämä johtuu siitä, että asiakasrajapinta mahdollistaa usean eri hakemisto-olion muokkaamisen yhdellä viestillä. Olioita ei kuitenkaan voida levittää yhdessä eteenpäin, jos ne kuuluvat replikointijärjestelmässä eri datajoukkoihin. Kahden eri datajoukon tietoja ei välttämättä haluta lähettää samoille hakemistopalvelimille.

5.5 Replikointi

Jotta muuttuvat osoitetiedot saataisiin leviämään kaikille palvelimille, on palveluun toteutettava replikointijärjestelmä. Replikoinnilla tarkoitetaan tässä yhteydessä tietokantapäivitysten kopiointia kullekin järjestelmän palvelimelle siten, että palvelun tiedot eri tietokannoissa ovat yhdenmukaisia. Replikointi perustuu monilähetykselle ja hoitaa osoitepalvelussa päivitysten proaktiivisen levityksen. Viestinnän ei kuitenkaan ole tarkoitus toimia täysin luotettavasti ja täten muun muassa viestikuittauksia ei käytetä. Replikoinnilla pyritään pitämään palvelimet samassa tilassa muttei taata sitä. Yhdenmukaisuuden takaamiseksi käytetään synkronointioperaatiota. Tässä osiossa kuvataan replikoinnin toiminnallisuus ja perustellaan järjestelmän viestintään ja rakenteeseen liittyvät valinnat.

5.5.1 Replikointivalinnat

Koska suunniteltavan osoitepalvelun on oltava hajautettu palvelu, ei yhden pääkopion replikointijärjestelmä ole käypä vaihtoehto. Hakemistotietoja on pystyttävä muuttamaan millä tahansa palvelimella, joten monen pääkopion replikointi on vaatimus. Tämä ratkaisu kuitenkin asettaa haasteita hakemistodatan yhdenmukaisuudelle.

Mikäli yhdenmukaisuus halutaan taata monen palvelimen järjestelmässä, on käytettävä tahdistettua replikointia. Se kuitenkin heikentää palvelun suorituskykyä merkittävästi. Etenkin haasteellisessa verkossa menetelmän käyttö muodostuu ongelmalliseksi. Tahdistus voi estää palvelun käyttämisen väliaikaisesti tai jopa lopullisesti. Tästä syystä ainoa toteuttamiskelpoinen ratkaisu on tahdistamaton replikointi ja lopulta saavutettava konsistenssi. Ne eivät kuitenkaan takaa hakemistotietojen yhdenmukaisuutta joka hetkellä. Näin ollen palveluun on toteutettava jonkinlainen mekanismi ristiriitojen hallintaa varten. Ajan myötä kaikki kopiot tulevat yhdenmukaistumaan. Tämä takaa palvelun jatkuvan toiminnan pitkilläkin aikaväleillä.

Replikoitavan datan voi jakaa kaikille palvelimille tai jaotella osiin. Haasteellisessa verkossa toimittaessa täyden replikoinnin toteuttaminen on ongelmallista muun muassa säilytystilan ja verkkokapasiteetin riittävyyden kannalta. Osittainen replikointi on mielekäs vaihtoehto resurssien säästämisen vuoksi, mutta sitä edellytetään kohdeverkossa myös tietoturvan takia. Kaiken hakemistodatan saatavuus yhdestä paikasta ei ole haluttua esimerkiksi tietomurtojen vuoksi. Osittainen replikointi kuitenkin asettaa haasteita muun muassa tiedon jakelulle, koska tällöin järjestelmä tarvitsee hakemistodatalle jakelupolitiikan.

Replikoitava data voidaan siirtää joko tiloina tai operaatioina. Koska operaatiot ovat tyypillisesti pienempiä, on niiden käyttö perusteltua kaista- ja yhteysaikarajoitteisissa verkoissa.

Hakemistopäivityksiin sovelletaan syntaktista ajastusta. Semanttinen ajastus olisi huomattavasti monimutkaisempi toteuttaa. Syntaktinen järjestelmä takaa lopulta saavutettavan konsistenssin ja on laskennallisesti yksinkertainen.

Tapahtumajärjestyksen määrittäminen tehdään reaaliaikaisilla kelloilla. Reaaliaikaleimat tarjoavat haasteelliseen verkkoon sopivan yksinkertaisen mutta riittävän tarkan ajastusmenetelmän. Vektorikellojen käyttö yhteysongelmista kärsivässä ja mahdollisesti hyvinkin suuressa verkossa ei ole toteuttamiskelpoista. Reaalikelloja käytettäessä eri laitteiden on suotuisaa olla kohtalaisen lähellä samaa aikaa, mutta varsinaista tahdistusvaatimusta ei ole. Lopulta saavutettavaan konsistenssiin on mahdollista päästä tahdistamattomillakin kelloilla.

Reaaliaikaisia kelloja käytettäessä ristiriitojen havaitseminen on vaivatonta tehdä syntaktisesti. Järjestelmä johtaa toisinaan olemattomien ristiriitojen havaitsemiseen mutta on huomattavasti semanttista ratkaisua kevyempi.

Mikään palvelin ei hylkää saapuvia päivityksiä, vaikka niiden ajo paikallisesti epäonnistuisikin. Koska palvelimet voivat hetkittäin olla eri tilassa, voi yhdessä paikassa epäonnistuva päivitys olla toisessa täysin validi. Syntaksiltaan vialliset päivitykset voidaan kuitenkin hylätä ja jättää levittämättä.

5.5.2 Yhdenmukaisuuden hallinta

Aikaleimat ja päivitystunnukset

Hakemistomuutokset nimetään yksilöllisillä tunnuksilla. Tunnukset koostuvat aikaleimasta, päivityksen tekijästä, palvelintunnuksesta sekä sekvenssinumerosta. Aikaleimaa käytetään hyödyksi operaatioiden suoritusjärjestyksen määrittämisessä. Muutoksen tekijän tunnuksella varmistetaan, että kaksi eri tahoa eivät voi luoda järjestelmään samannimisiä päivityksiä. Palvelintunnisteen tarkoitus on taata, etteivät eri palvelimet voi luoda samannimistä päivitystä. Sekvenssinumerolla varmistutaan vielä siitä, että kaksi samalla laitteella tehtyä päivitystä saavat eri tunnukset, vaikka ne luotaisiin käyttäen samaa aikaleimaa. Unix-aikaa käyttävä tunnus näyttäisi siis tältä: `1363006862-user1000-server100-1`.

Aikaleimat luodaan ensimmäisellä päivitystä käsittelevällä palvelimella. Palvelinten kellojen ei tarvitse olla tarkasti tahdistettuja, mutta se parantaa hakemistodatan oikeellisuutta. Aikaleimojen pääasiallinen tarkoitus on taata eri palvelinten hakemistotietojen konvergoituminen, ei muutosten todellista aikajärjestystä.

Koska päivitysten tunnukset ovat uniikkeja, voidaan samannimisten päivitysten olettaa olevan sisällöltään samat. Uudelleen saapuvat päivitykset voidaan siis automaattisesti hylätä. Lisäksi päivityksiä vertaillessa eri palvelinten kesken riittää, että verrataan päivitystunnuksia eikä varsinaista sisältöä. Tämä vähentää tarpeetonta laskentaa järjestelmässä.

Takaisinpaluu

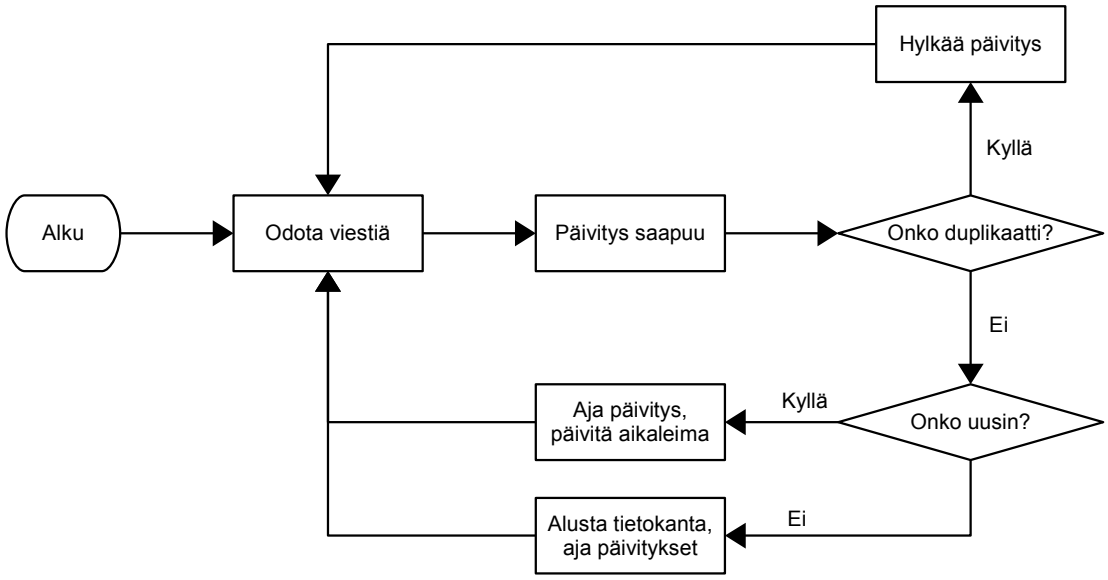
Haastellisessa verkossa hakemistopäivitykset saattavat saapua palvelimelle tarkoitetusta suoritusjärjestyksestä poikkeavasti. Koska operaatioiden suoritusjärjestys voi vaikuttaa hakemiston tilaan, on yhdenmukaisuuden takaamisen vuoksi varmistettava, että päivitysten ajojärjestys on oikea. Palvelin päättelee oikean ajojärjes-

tyksen aikaleimoista. Mikäli saapuvalla päivityksellä on vanhempi aikaleima kuin viimeksi suoritetulla päivityksellä, tiedetään ajojärjestyksen olleen väärä.

Väärän ajojärjestyksen sattuessa palvelimen on palattava taaksepäin päivityshistoriassa. Koska yksittäisiä päivityksiä ei voida helposti kumota, suoritetaan hakemistolle takaisinpaluu. Käytännössä tämä tarkoittaa, että hakemiston tiedot alustetaan uudestaan. Tällöin olemassa oleva hakemistodata poistetaan ja alkuperäinen sisältö ladataan palvelimelle. Tämän jälkeen kaikki saapuneet päivitykset ajetaan uudestaan aikaleimojen määräämässä järjestyksessä. Takaisinpaluuta ei ole välttämätöntä suorittaa välittömästi, vaan se voidaan ajastaa myöhempään ajankohtaan. Tämä menetelmä vähentää takaisinpaluuoperaatioiden määrää, mikäli palvelimelle saapuu peräkkäin useita päivityksiä väärässä järjestyksessä. Toisaalta tällöin hakemistojen yhdenmukaistuminen luonnollisesti viivästyy.

Mikäli järjestelmässä on menetelmä todeta hakemistojen konsistenssi, voidaan takaisinpaluuta nopeuttaa. Tällöin yhdenmukaisesta hakemistotilasta otetaan kopio, jota käytetään tietoja alustettaessa. Näin voidaan vähentää uudelleen ajettavien päivitysten määrää. Tässä toteutusmallissa tarvittavaa menetelmää ei kuitenkaan ole, joten alustaminen suoritetaan aina lähtötietojen pohjalta.

Kuva 5.4 esittää takaisinpaluumekanismin toiminnan. Alussa hakemistopalvelin odottaa päivityksiä. Kun sellainen saapuu verkosta, tarkistaa ohjelma, onko samainen päivitys saatu jo aikaisemmin. Mikäli uusi päivitys osoittautuu duplikaatiksi, se voidaan hylätä. Palvelin palautuu jälleen odottamaan uusia päivityksiä. Jos saapunutta muutosta ei ole saatu aikaisemmin, täytyy se käsitellä. Saapunut päivitys voidaan suorittaa, mikäli se on uudempi kuin uusin hakemistoon ajettu päivitys. Jos hakemistossa on jo saapunutta päivitystä uudempia muutoksia, täytyy takaisinpaluu suorittaa. Tietokanta siis alustetaan uudestaan ja päivitykset suoritetaan niiden oikeassa järjestyksessä.



Kuva 5.4: Hakemistopalvelimen takaisinpaluumekanismin tilakaavio.

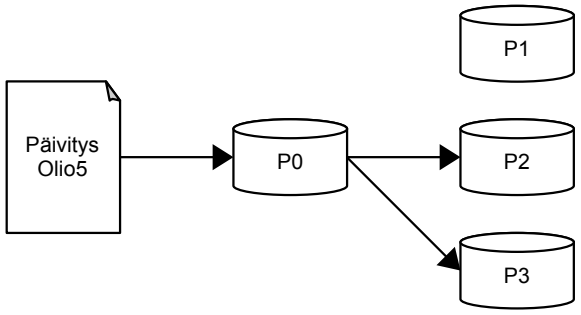
5.5.3 Tilausryhmät ja datajoukot

LDAP-palvelinten pääsynhallintalistoja voidaan käyttää hakemistossa sijaitsevan tiedon leviämisen rajoittamiseen. Koska itse päivitykset eivät ole varsinaista hakemistodataa, on replikointi erillään LDAP-palvelimesta. Näin ollen OpenLDAPin pääsynhallintaa ei voida soveltaa päivitysten levittämiseen, vaan ainoastaan niiden suorittamiseen paikallisella palvelimella. Koska kaikkia päivityksiä ei kuitenkaan saa jakaa kaikille palvelimille, on replikointijärjestelmää varten tehtävä erillinen pääsynhallintamekanismi. Tähän tarkoitukseen käytetään tilausryhmiä. Tilausjärjestelmä toimii eräänlaisena palvelinten välisenä postituslistana. Ryhmien avulla voidaan määrittää, mihin hakemistopäivitykset pitää välittää.

Ennen hakemiston hajauttamista palveluun määritellään datajoukot, joihin kaikki hakemistodata jaetaan. Datajoukon muodostaa hakemisto-olioista koostuva ryhmä, joka voidaan määritellä esimerkiksi olioiden tunnisteiden tai hakemistohierarkian avulla toteutuksen tarpeista riippuen. Kaikki kyseisiin olioihin kohdistuvat muutokset lähetetään datajoukon tilanneille osoitepalvelimille. Joukko- ja vastaanottajatiedot voidaan tallentaa esimerkiksi itse hakemistopalvelimelle. Nämä tiedot määritellään tässä mallissa palveluun pääsääntöisesti ennen järjestelmän hajautusta, mutta niiden muuttaminen jälkikäteen on mahdollista. Esimerkki levitysmekanismin toimintaperiaatteesta näkyy kuvassa 5.5.

Datajoukkomääritelmät (P0)	
Tietoalkiot	Datajoukko
Olio1, Olio2...	A
Olio3, Olio4...	B
Olio5, Olio6...	C
...	...

Vastaanottajalista (P0)	
Datajoukko	Vastaanottajat
A	P1, P2, P3...
B	P1, P2...
C	P2, P3...
...	...



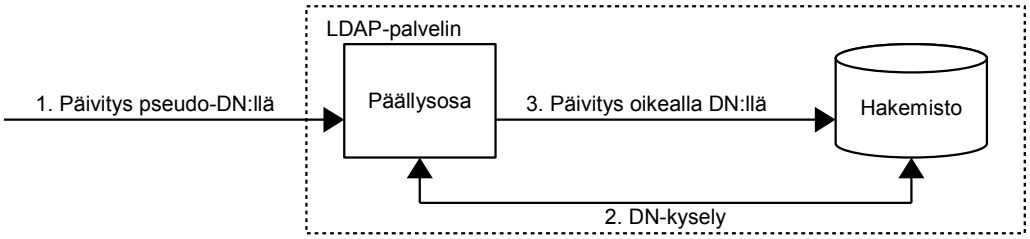
Kuva 5.5: Päivitysten replikoinnissa käytetty mekanismi: yllä palvelimen P0 datajoukot sekä vastaanottajalistat, alla esimerkki päivityksen välityksestä verkkoon.

5.5.4 Hakemisto-olioihin viittaaminen

LDAPissa olioihin viitataan niiden DN-nimellä, joka käytännössä ilmaisee sijaintia hierarkiassa. Jos olioita siirrellään hakemiston sisällä, DN-nimet muuttuvat. Tällöin esimerkiksi olion attribuutin muuttaminen epäonnistuu, mikäli oliota ei ole vielä siirretty kaikilla palvelimilla ja viittaus tehdään uudella nimellä.

Ongelma voidaan välttää viittaamalla olioihin esimerkiksi jollakin attribuuttiarvolla. Tällöin on vain varmistuttava, että arvo on uniikki, jotta muutokset tehdään oikeaan olioon. Attribuutin käyttäminen tunnisteena helpottaa myös olioihin viittaamista, koska koko DN-nimen muistaminen ei ole tarpeellista. OpenLDAPissa attribuutilla viittaaminen on mahdollista tehdä käyttämällä rwm-päällysosaa [25]. Se toimii LDAP-viestien datan uudelleenkirjoittajana ja pystyy tekemään muutoksen attribuuttiarvosta olion DN-nimeen. Käytännössä päällysosa suorittaa ylimääräisen LDAP-haun, jolla olion DN selvitetään. Etuna on kuitenkin se, ettei DN-nimeä tarvitse tietää etukäteen. Pseudo-DN:n uudelleenkirjoitus todelliseksi DN:ksi on esitetty kuvassa 5.6.

Uudelleenkirjoitustoiminnallisuus on mahdollista tehdä myös ilman päällysosaa. Tällöin LDAP-palvelimen kanssa asioivan sovelluksen on kuitenkin itse tehtävä DN:n selvittävä haku ennen haluttua hakemisto-operaatiota.

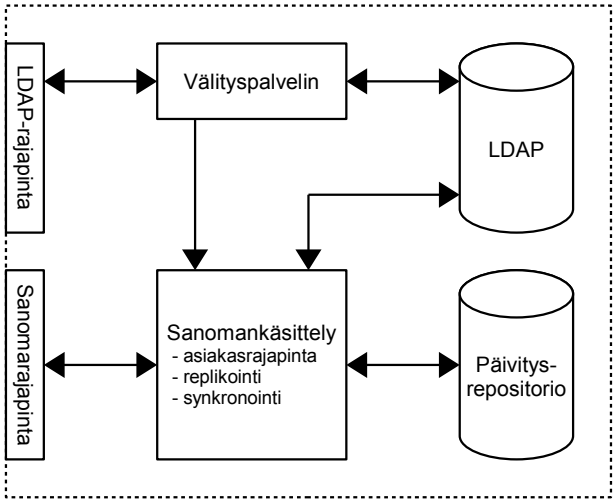


Kuva 5.6: Pseudo-DN:n muuttaminen oikeaksi DN:ksi päivityksen saapuessa.

5.5.5 LDAP-päivitysten replikointi

Mikäli LDAP-rajapinnalla halutaan tukea myös kirjoitusoperaatioita, on replikointijärjestelmää laajennettava. Myös suoraan LDAPin välityksellä saapuvat päivitykset on välitettävä muille palvelimille, jotta yhdenmukaisuus saadaan säilytettyä.

LDAP-operaatioihin päästään käsiksi toteuttamalla välityspalvelin. Sen tarkoitus on lukea LDAPin kautta saapuvat kyselyt ja lähettää hakemistopäivitykset replikoitaviksi. Välityspalvelin tutkii saapuvia LDAP-viestejä ja päästää haut läpi sellaisenaan. Muutospyynnöistä luetaan sisältö, minkä pohjalta tehdään replikoitava ja päivitysrepositorioon tallennettava merkintä. LDAP-päivitys voidaan muuttaa esimerkiksi LDIF-muotoon. Kuvassa 5.7 on esitetty yksi malli LDAP-välityspalvelimen sijoittamiseksi osoitepalvelimelle.



Kuva 5.7: LDAP-välityspalvelimen sijoittuminen osoitepalvelimen sisäiseen arkkitehtuuriin.

5.6 Synkronointi

Koska replikointijärjestelmän viestintä ei toimi luotettavasti, voi hakemistopäivityksiä jäädä siirtämättä palvelimille verkko-ongelmien takia. Synkronointijärjestelmän tarkoitus on palauttaa eri hakemistojen välinen yhdenmukaisuus. Synkronointia tarvitaan esimerkiksi tilanteessa, jossa verkko on jakautunut osiin ja eri saarekkeissa on tehty päivityksiä. Yhteyden palatessa verkkonoodien on vaihdettava katkon aikana tehdyt päivitykset keskenään, jotta yhdenmukaisuus saavutetaan. Synkronointi on siis reaktiivinen toimenpide. Synkronointijärjestelmä käyttää samaa päivitysformaattia kuin replikointijärjestelmäkin. Tarvittaessa myös synkronointi voi käynnistää hakemistolle takaisinpaluun.

Haasteellisessa verkossa synkronoinnissa on otettava huomioon toimintaympäristön rajoitteet. Synkronointi on pyrittävä suorittamaan mahdollisimman pienellä viestimäärällä, sillä useiden viestien käyttäminen pidentää viestinnän kestoa. Hitailla yhteyksillä kestot voivat olla merkittäviä. Epäluotettavilla yhteyksillä suurempi viestimäärä aiheuttaa myös luonnollisesti enemmän uudelleenlähetystarvetta. Lisäksi koko synkronointioperaatio voi jäädä kesken, jos verkko jakautuu uudestaan. Toisaalta synkronoinnissa on pyrittävä välttämään tarpeettoman tiedon siirtoa, joten esimerkiksi kaikkien päivitysten lähettäminen kerralla ei ole käytännöllinen malli. Viestien määrän ja koon välille on löydettävä ratkaisu, joka toimii luotettavasti kaavaillussa toteutusympäristössä.

Toisin kuin replikoinnin, synkronoinnin on toimittava luotettavasti. Jotta operaation voidaan taata menevän onnistuneesti loppuun, on viestit kuitattava. Tässä mallissa ei uudelleenlähetysajastimia tai viestikuittauksia ole kuitenkaan käytetty. Alempien verkkokerrosten oletetaan tarjoavan kyseiset toiminnallisuudet. Mikäli mekanismeja ei ole tarjolla, on niiden lisääminen sovelluskerrokselle mahdollista.

5.6.1 Viestintämalli

Kahden palvelimen vertaillessa päivitystietojaan niiden on kerrottava toisilleen, mitä päivityksiä ne ovat saaneet. Vasta tämän jälkeen puuttuvat tiedot voidaan lähettää. Synkronointiviestintä koostuu neljästä eri viestityypistä, jotka on listattu alla. Kaksi ensimmäistä keskittyvät tiedon vertailuun ja toiset kaksi tiedon vaihtamiseen. Viestiformaatiksi voidaan replikoinnin tapaan valita mikä tahansa LDIF-tiedon siirtoon sopiva muoto.

- Hash Log - Viestin lähettäjä kertoo vastaanottajalle eri aikaväleistä koostettujen päivitystietojen tarkistussummat sekä aikavälitiedot.

- Match Log - Tarkistussummia vertailtuaan Hash Login vastaanottanut osapuoli kertoo, miltä osin laitteiden päivitystiedot täsmäävät ja eroavat. Eroavilta aikaväleiltä listataan kaikki päivitykset.
- Update Request - Puuttuvia päivityksiä voidaan pyytää tällä viestillä.
- Update - Tällä viestillä siirretään päivityksiä laitteelta toiselle.

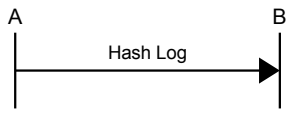
Päivitystietoja vertailtaessa kulunut aika jaetaan jaksoihin. Kunkin jakson päivityksiä käsitellään yhtenä ryhmänä. Aluksi kullekin aikavälille lasketaan tarkistussumma, joka lähetetään vastapuolelle. Toinen palvelin laskee oman summansa samoille aikaväleille ja suorittaa vertailun. Mikäli tarkistussummat eroavat, tiedetään kyseisten aikavälien päivitysryhmien eroavan palvelinten välillä.

Itse aikavälit voidaan valita miten tahansa. Yksi tapa on käyttää logaritmista jakoa, jossa vanhemmat aikavälit ovat pidempiä. Tämän valinnan pitäisi vähentää liikennettä olettaen, että vanhemmat aikavälit ovat uusia yhdenmukaisempia aikaisemmin suoritettujen synkronointien johdosta.

Tarkistussummia vertailtuaan palvelimet voivat vaihtaa eroavien aikavälien osalta päivitysten tunnuksia, joiden pohjalta puuttuvia tietoja voidaan siirtää. Viestimäärän vähentämiseksi yhdessä sanomassa voidaan lähettää kaksi viestiä. Käytännössä tällainen tilanne tulee vastaan, kun palvelin lähettää tietojaan toiselle ja pyytää samalla itseltään puuttuvia tietoja.

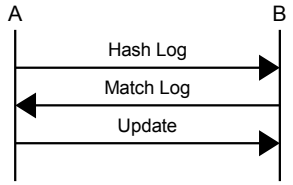
Tässä viestintämallissa synkronointikeskustelu voidaan käydä neljällä eri tavalla riippuen siitä, mitä tietoja kullakin noodilla on. Alla esitetyissä tapauksissa noodit A ja B pyrkivät synkronoimaan tietonsa. A on keskustelun aloittava osapuoli. A:n tietojoukkoa merkitään U_a:lla ja B:n U_b:llä, käsiteltävät päivitysjoukot ovat X, Y ja Z.

Kuva 5.8 esittää tapauksen, jossa noodeilla A ja B on samat tietojoukot. Toisin sanoen pätee, että $U_a = U_b$. Osapuoli A aloittaa viestinnän lähettämällä B:lle tarkistussummat. B havaitsee A:n summien täsmäävän omiinsa ja päättelee molemmilla olevan samat tiedot. Lisäviestintää ei siis tarvita.



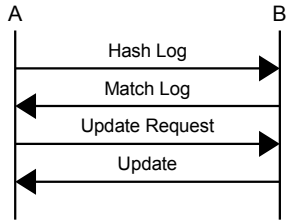
Kuva 5.8: Synkronointi tapauksessa, jossa vaihdettavaa tietoa ei ole.

Kuvassa 5.9 näkyy tilanne, jossa viestinnän aloittavalla osapuolella on toiselta laitteelta puuttuvia päivityksiä. Tietojoukot tässä tapauksessa ovat siis $U_a = \{X \cup Y\}$ ja $U_b = \{X\}$. A aloittaa viestinnän lähettämällä tarkistussummat B:lle. B huomaa summien eroavan ja ilmoittaa asiasta Match Log -viestillä, josta A näkee eroavat aikavälit sekä eroavien aikavälien päivitysten tunnukset. Koska A:lla on kaikki B:n omistamat päivitykset, se ei pyydä B:tä lähettämään mitään. Se kuitenkin vastaa kyselyyn lähettämällä B:ltä puuttuvat tiedot Update-viestillä. Tämän seurauksena tietojoukot lopussa ovat $U_a = U_b = \{X \cup Y\}$. Synkronointi suoritettiin kolmella viestillä.



Kuva 5.9: Tietojen synkronointi A:lta B:lle.

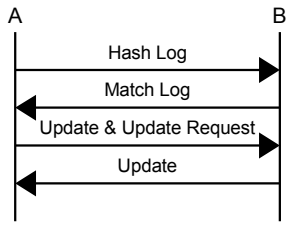
Tilanne, jossa tietojoukot ovat edelliseen tapaukseen nähden toisin päin, on esitetty kuvassa 5.10. Joukot voidaan kuvata lauseilla $U_a = \{X\}$ ja $U_b = \{X \cup Y\}$. Laitteiden välinen viestintä alkaa A:n lähettämällä Hash Log -viestillä. B havaitsee, etteivät tarkistussummat täsmää, joten se vastaa edellisen tapauksen mukaisesti Match Log -viestillä. A huomaa Match Logista, että B:llä on kaikki A:lta löytyvät päivitykset. A pyytää puuttuvia tietoja B:ltä käyttämällä Update Requestia, johon se saa vastauksena Update-viestin. Synkronointiin vaadittiin neljä viestiä ja lopputilanteessa pätee $U_a = U_b = \{X \cup Y\}$.



Kuva 5.10: Tietojen synkronointi B:ltä A:lle.

Mikäli molemmille verkkolaitteille on saapunut uutta tietoa ennen synkronoinnin aloittamista, täytyy päivityksiä viestiä kumpaankin suuntaan. Tilanne näkyy kuvassa 5.11. Alkutilanteessa pätee $U_a = \{X \cup Y\}$ ja $U_b = \{X \cup Z\}$. Synkronointi

alkaa normaalisti A:n lähettämällä Hash Log -viestillä. B vastaa ilmoittamalla tarkistussummat ja antamalla Match Log -listauksen. Tästä listasta A voi päätellä itseltään ja vastapuolelta puuttuvat päivitykset. Vastauksena lähetetäänkin nyt yhdistelmäviesti, jossa on sekä Update Request että Update. Update Requestillä pyydetään puuttuvia päivityksiä ja Updatellä B:lle lähetetään siltä puuttuvat tiedot. Lopuksi B vastaa Update Requestiin Update-viestillä. Lopputilanteessa pätee $U_a = U_b = \{X \cup Y \cup Z\}$. Synkronointi vaati neljä viestiä.



Kuva 5.11: Tietojen synkronointi molempiin suuntiin.

5.6.2 Verkon synkronointi

Synkronointi käynnistetään, kun aikaisemmin katkennut kahden laitteen välinen yhteys palaa toimintaan. Operaation aloittava taho voidaan päättää esimerkiksi laitteiden tunnusten perusteella. Näin vältytään kahdelta päällekkäiseltä synkronointioperaatiolta. Tämä mekanismi edellyttää, että keino palautuvan yhteyden havaitsemiseksi on saatavilla. Kohdeverkossa tämä onnistuu naapureiden kesken, mutta yhteyden tilasta muihin noodeihin ei saada tietoa.

Tietoja synkronoidaan myös joissakin muissa tapauksissa. Saatuaan päivityksiä yhdeltä naapuriltaan laitteet käynnistävät uuden synkronointioperaation muiden naapureittensa kanssa. Näin tieto leviää verkossa laite kerrallaan. Tarpeetonta viestintää voi kuitenkin syntyä topologioissa, joissa noodeilla on yhteisiä naapureita.

Näillä toiminnoilla tietoa ei kuitenkaan saada siirrettyä muille kuin naapureille. Tästä johtuen päivityksiä ei voida välittää, mikäli kahden palvelimen välissä on noodi, joka ei ole kiinnostunut samoista tietojoukoista. Tällaisia tapauksia varten voidaan järjestelmään toteuttaa esimerkiksi poikkeuslista, joka määrittää naapureiden yli tehtävät synkronoinnit. Näitä synkronointeja voidaan ajaa vaikkapa tasaisin väliajoin.

5.7 Tietoturva ja pääsynhallinta

Tietojen hajauttamisen lisäksi on syytä huolehtia tietoturvasta myös muilla tavoin. Näin voidaan tehdä esimerkiksi salaamalla päivitysten sisältö. Lisäksi osoitepalvelun on noudatettava jonkinlaista pääsynhallintapolitiikkaa, jotta tietojen leviäminen väärille tahoille voidaan estää. OpenLDAP tarjoaa monipuolisen pääsynhallintajärjestelmän. Pääsynhallintalistoilla (ACL, Access Control List) voidaan määritellä sääntöjä, joiden mukaan hakemiston käyttäjille jaetaan oikeuksia dataan. [23]

Pääsylistoja luodessa yhteen sääntöön määritellään kohdedata, käyttäjäryhmä sekä oikeustaso. Kohdedataa ja käyttäjäryhmää kuvatessa voidaan käyttää hyväksi useita erilaisia menetelmiä. Näihin lukeutuvat muun muassa joukko-oppi, regexlausekkeet, käyttäjäryhmät, hakemistohierarkia sekä IP-osoitteet. Myös oikeustasoja on useita. Tyypillisimmät niistä ovat luku- ja kirjoitusoikeudet mutta oikeudet voidaan rajata myös esimerkiksi autentikointiin tai vertailuun. [23]

Kaikki hakemistoon tehdyt operaatiot ajetaan pääsynhallinnan läpi. Tämä koskee sekä kyselyitä että muutoksia. Näin taataan, että ainoastaan sallitut käyttäjät voivat hakea tai muokata tietoja.

Koska sanoma- ja IP-verkoille halutaan erilaiset pääsynhallintasäännöt, on eri verkoista tulevat kyselyt kyettävä erottamaan toisistaan. Hakemistopalvelimen pääsynhallintajärjestelmä voi erottaa eri verkoista tulevat yhteydet esimerkiksi IP-osoitteen avulla. Sanomaverkon kautta saapuneet kyselyt jäävät itse palvelimen suoritettavaksi. Toisin sanoen sanomakyselyt tulevat hakemistoon localhost-osoitteesta. IP-verkosta tulevat hakemistopyynnöt tulevat suoraan palvelinlaitteen ulkopuolelta, joten ne käyttävät ulkoista IP-osoitetta.

Pääsynhallintaa on rajoitettava siten, että osoitekyselyitä voidaan tehdä sekä sanoma- että IP-verkosta. Muutokset osoitetietoihin kuitenkin sallitaan ainoastaan sanomaverkon kautta. Tämä helpottaa esimerkiksi replikointijärjestelmän toteutusta, koska IP-verkon kautta tulevia muutospyyntöjä ei tarvitse käsitellä ja muuttaa replikointiviesteiksi.

Kun osoitepalvelua käytetään IP-verkossa, ottaa asiakas suoran LDAP-yhteyden osoitepalvelimelle. Käyttäjä kirjautuu hakemistoon omilla tunnuksillaan, jolloin LDAP-palvelimen pääsynhallintasäännöt rajoittavat tietojen saatavuutta.

Sanomaverkossa käyttäjien ja laitteiden autentikoinnin oletetaan tapahtuvan alemmilla verkkokerroksilla. Osoitepalvelun ei siis tarvitse itse toteuttaa käyttäjän todennusta. Hakemistopalvelin saa muutos- tai kyselyviestissä sanoman lähettäjän tunnuksen, jolla se kirjautuu hakemistoon LDAPin avulla. Tällöin lopullisen pääsynhallinan tekevät LDAP-palvelimen ACL-säännöt. Yksinkertaista kirjautumista

käytettäessä LDAP-palvelimelle autentikoituminen vaatii käyttäjätunnuksen lisäksi salasanan [23]. Koska itse osoitepalvelu ei käytä salasanoja, täytyy palvelimelle luoda valesalasana-attribuutti, jolla kirjautuminen suoritetaan.

Luku 6

Synkronointimallin simulointi

Ehdotetusta synkronointijärjestelmästä tehtiin toteutus, jonka avulla mallin toiminnallisuutta ja tehokkuutta voitiin testata. Tässä luvussa esitellään kyseisen järjestelmän simuloinnista kerätyt mittaustulokset sekä analysoidaan näitä tuloksia.

6.1 Koeympäristö

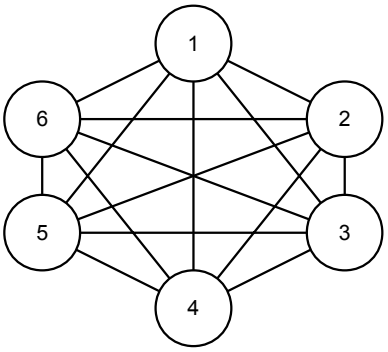
Luvussa 5 esitettyä mallia synkronointijärjestelmälle haluttiin testata erilaisissa verkkoympäristöissä. Tätä varten järjestelmästä tehtiin toteutus, jota voitiin ajaa simuloidussa verkossa eri topologioilla. Koeverkkoa simuloitiin ns-3 -verkkosimulaattorilla. Noodeina toimivat Linux Containereissa ajetut synkronointiohjelmat.

Mittausten tarkoituksena oli selvittää, kuinka paljon liikennettä järjestelmä synnyttää. Liikenne mitattiin viestien lukumäärässä eikä siirretyssä datassa, sillä viestimäärä on esitetylle mallille ominainen, kun taas datamäärä riippuu toteutuksesta sekä siirrettävästä tiedosta.

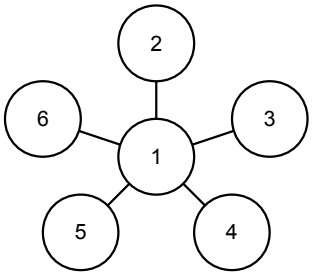
Koska synkronointimallissa verkkonoodit eivät koordinoi tiedonsiirtoa keskenään, voi verkkoon syntyä kilpatilanteita. Tällöin yhdellä noodilla saattaa olla useampi synkronointioperaatio käynnissä samaan aikaan. Lopullinen viestimäärä riippuu siis merkittävästi siitä, missä järjestyksessä palvelimet viestivät. Simulaatio nähtiin hyvänä tapana selvittää, kuinka verkko käyttäytyisi todellisuudessa.

Mittaukset suoritettiin neljällä eri verkkotopologialla, jotka kaikki koostuivat kuudesta verkkonoodista. Ainoastaan noodien välisiä yhteyksiä vaihdeltiin. Valitut topologiat olivat täysin kytketty verkko (kuva 6.1), tähti (kuva 6.2), rengas (kuva 6.3) ja ketju (kuva 6.4).

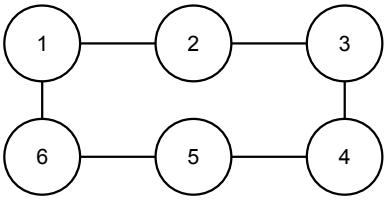
Täysin kytketyllä verkolla voidaan tutkia, kuinka tehokkaasti palvelimet yhdenmukaistuvat, kun noodeilla on useita yhteisiä naapureita. Tähtimallinen verkko näyttää, kuinka paljon liikennettä palvelimille syntyy, kun yksi noodi yhdistää kaikkia muita. Rengasverkolla voidaan tutkia tiedon etenemistä suljetussa lenkissä. Ketjutopologia puolestaan kertoo, miten liikenne jakautuu, kun päädyissä olevat noodit voivat välittää tietoa vain useiden muiden noodien lävitse.



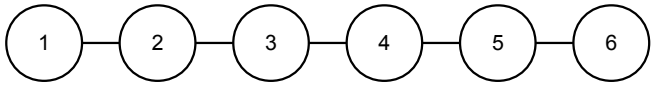
Kuva 6.1: Kokeessa käytetyn täysin kytketyn verkon topologia.



Kuva 6.2: Kokeessa käytetyn tähtiverkon topologia.



Kuva 6.3: Kokeessa käytetyn rengasverkon topologia.



Kuva 6.4: Kokeessa käytetyn ketjuverkon topologia.

Simuloinnin tarkoituksena oli tutkia, kuinka paljon viestiliikennettä syntyy, kun verkkonoodit synkronoivat tietonsa. Alkutilanteesta haluttiin tehdä haastava, jotta nähtäisiin, miten hyvin järjestelmä selviää ääritilanteesta. Tätä varten kullekin palvelimelle annettiin oma päivitystiedostonsa, jota muilla noodeilla ei ollut. Jokaisella palvelimella oli siis dataa annettavana kaikille muille ja saatavana kaikilta muilta. Lopputilanteessa kaikilla noodeilla oli kaikki tiedot.

Kunkin simulaatioajon alussa noodien väliset verkkoyhteydet olivat poikki. Kaikki yhteydet palautettiin samaan aikaan, jotta palvelinten välille saataisiin mahdollisimman monta kilpatilannetta. Kutakin verkkotopologiaa simuloitiin kymmenellä ajolla. Jokaisen ajon jälkeen kunkin noodin lähettämien ja vastaanottamien viestien lukumäärät kirjattiin ylös.

6.2 Tulokset ja analyysi

Mittausten tulokset näkyvät liitteessä A. Kunkin verkkotopologian mittaukset on jaettu kahteen taulukkoon: lähetetyt ja vastaanotetut viestit. Simulaatioista laskettiin palvelinten viestimäärille myös keskiarvot ja keskihajonnat.

Täysin kytketty topologia

Täysin kytketty verkko oli simulaatioista järjestelmälle haastavin. Kullakin noodilla oli tässä testitapauksessa viisi naapuria, mikä aiheutti merkittävän määrän kilpatilanteita. Lähetetyt viestit näkyvät taulukossa A.1 ja vastaanotetut taulukossa A.2. Pahimmillaan tietojen yhdenmukaistamiseen vaadittiin yli 150 viestiä ja parhaimmillaan hieman yli 60. Tämä tarkoittaa sitä, että järjestelmä voi toimia toisinaan hyvin epäoptimaalisesti. Pienellä koordinoinnilla viestimäärää voidaan saada pudotettua merkittävästi.

Kussakin ajossa oli usein yksi noodi, joka joutui viestimään muita enemmän. Huomattavaa on myös, että vaikka alkutilanne verkossa oli palvelinten kannalta sym-

metrinen, oli noodin 5 keskimääräinen viestimäärä muita isompi. Tämä saattaa johtua esimerkiksi simulaatioverkon ominaispiirteistä kuten linkkien käynnistysjärjestyksestä. Kyseinen noodi ei kuitenkaan poikkeuksetta ollut ajojen eniten viestinyt palvelin.

Tähtitopologia

Tähtiverkon mittaustulokset näkyvät taulukoissa A.3 ja A.4. Topologian keskellä ollut noodi joutui luonnollisesti viestimään eniten. Ympärillä olleiden palvelinten viestimäärät pysyivät melko alhaalla ja hajontakin oli vähäisempää. Myös koko verkon yhteenlaskettu viestimäärä vaihteli suhteellisen vähän. Tyypillisesti kaikkien noodien synkronointi vaati noin 50 viestiä, mutta parissa tapauksessa määrässä nähtiin piikki. Kokonaisviestimäärältään tämä topologia oli simulaatioista tehokkain.

Rengastopologia

Rengastopologia oli täysin kytketyn verkon lisäksi ainoa, jonka noodiasetelma oli symmetrinen. Simulaatioiden tulokset on listattu taulukoissa A.5 ja A.6. Tässä verkossa eri noodien viestimäärät olivat keskiarvoltaan melko lähellä toisiaan. Merkittävää tässä verkkomallissa oli kuitenkin kokonaisviestimäärän varsin suuri keskihajonta.

Ketjutopologia

Viimeisenä koetopologiana verkossa oli ketjumalli, jonka tulokset näkyvät taulukoissa A.7 ja A.8. Ketjun päädyissä olevat noodit viestivät palvelimista vähiten. Tähän vaikutti luonnollisesti se, ettei noodeilla ollut kuin yksi naapuri, mistä johtuen kilpatilanteita ei syntynyt. Koko verkon viestimäärän keskihajonta oli samaa luokkaa, kuin rengastopologiassa, mutta itse viestimäärä pysyi alhaisempana.

Mallin jatkokehitys

Simuloinnilla saatiin osoitettua esitetyn synkronointimallin toimivan käytännössä. Mallin havaittiin kuitenkin olevan liikennemäärältään epäoptimaalinen haastavissa tilanteissa. Koska viestimäärä haasteellisessa verkossa halutaan pitää mahdollisimman pienenä, on mallia syytä vielä kehittää. Liikenteen määrää voidaan optimoida eri tavoin.

Kilpatilanteita voidaan vähentää merkittävästi muun muassa käyttämällä ajastimia ja rajoittamalla samanaikaisten päivitysoperaatioiden määrää. Tällöin kukin noodi käynnistäisi esimerkiksi vain yhden synkronointioperaation kerrallaan ja käynnistäisi seuraavan vasta edellisen loputtua tai viiveajastimen umpeuduttua.

Monimutkaisempi tapa vähentää liikennettä on koordinoida viestintää noodien kesken. Tässä menetelmässä palvelimet voisivat vaihtaa noodien tila- tai naapuruustietoja. Näin palvelimet saisivat tietoaan ympäristöstään ja voisivat karsia pois tarpeettomia synkronointioperaatioita.

Liikenteen määrää voidaan optimoida myös laitteiden synkronointiohjelmiston täsmällisellä konfiguroinnilla. Tämä menetelmä mahdollistaa synkronointiliikenteen mukauttamisen tarkalleen verkon tarpeisiin. Toisaalta konfigurointi tuo verkon ylläpitoon monimutkaisuutta ja altistaa järjestelmän toiminnan käyttäjän tekemille asetusvirheille.

Luku 7

Yhteenveto

Tämän työn tavoitteena oli suunnitella hajautettu osoitepalvelu haasteelliseen verkkoon. Suunnittelu aloitettiin tutustumalla haasteellisiin verkkoihin ja niiden ominaispiirteisiin sekä rajoitteisiin. Tarkoituksena oli selvittää, minkälaisessa ympäristössä osoitepalvelun pitäisi toimia. Tämän jälkeen työssä otettiin tarkasteluun erilaisia osoitepalveluiden pohjana käytettyjä teknologioita. Vertailua suoritettiin sekä protokollien että ohjelmistototeutusten kesken.

Työssä perehdyttiin myös erilaisiin tietomalleja ja -liikennettä koskeviin vaihtoehtoihin. Näiden läpikäyminen oli olennaista, jotta suunniteltu järjestelmä ottaisi huomioon alla olevan verkon rajoitteet. Niiden selvitysten pohjalta osoitepalvelulle esitettiin ehdotettu toteutusmalli. Malli kattoi teknologiavalinnat, viestintärajapinnat sekä replikointi- ja synkronointimenetelmät. Lopuksi suoritettiin synkronointimallin simulointi ja mittauksista saatujen tulosten analysointi.

Simuloinnilla osoitettiin esitetyn synkronointimallin olevan toimiva mutta tarvitsevan optimointia. Järjestelmän jatkokehitys voitaisiin kohdistaa esimerkiksi viestimäärän pienentämiseen. Tämä vaatisi erilaisten optimointimenetelmien tutkimista ja testaamista.

Viitteet

[1] Bernstein, P. A., Hadzilacos, V., ja Goodman, N. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[2] Butcher, M. *Mastering OpenLDAP*. Packt Publishing Ltd., 2007.

[3] Carter, G. *LDAP System Administration*. O'Reilly, 2003.

[4] Clemm, G., et al. Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol. RFC 4918 (Proposed Standard), toukokuu 2004.

[5] Craig, M., Forgerock AS. OpenDJ 2.6.1 Release Notes. `http://docs.forgerock.org/en/opendj/2.6.1/OpenDJ-2.6.1-Release-Notes.pdf`. Viitattu 5.3.2014.

[6] Daboo, C. CardDAV Directory Gateway Extension. Internet Draft, elokuu 2010.

[7] Daboo, C. CardDAV: vCard Extensions to Web Distributed Authoring and Versioning (WebDAV). RFC 6352 (Proposed Standard), elokuu 2011.

[8] Demmer, M., Du, B., ja Brewer, E. TierStore: A Distributed Filesystem for Challenged Networks in Developing Regions. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2008), FAST'08, USENIX Association, s. 3:1–3:14.

[9] Donley, C. *LDAP Programming, Management and Integration*. Manning, 2003.

[10] Dusseault, L. *WebDAV: Next-Generation Collaborative Web Authoring*. Prentice Hall PTR, 2003.

[11] Dusseault, L. HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV). RFC 4918 (Proposed Standard), kesäkuu 2007.

[12] Fall, K. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2003), SIGCOMM '03, ACM, s. 27–34.

[13] Gilbert, S., ja Lynch, N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *SIGACT News 33*, 2 (kesäkuu 2002), 51–59.

[14] Good, G. The LDAP Data Interchange Format (LDIF) - Technical Specification. RFC 2849 (Proposed Standard), kesäkuu 2000.

[15] Hoffer, J. A., Prescott, M. B., ja McFadden, F. R. *Modern Database Management*, eighth ed. Pearson Prentice Hall, 2007.

[16] IBM. IBM Tivoli Directory Server. `ftp://public.dhe.ibm.com/software/tivoli/datasheets/TID10444-USEN-00.pdf`. Viitattu 26.2.2013.

[17] International Organization for Standardization. Information technology - Syntactic metalanguage - Extended BNF. ISO/IEC 14977, 1996.

[18] Kivinen, J., Helsingin yliopisto, Tietojenkäsittelytieteen laitos. Tietorakenteet. `https://www.cs.helsinki.fi/u/jkivinen/opetus/tira/k08/kaikki.pdf`, 2008.

[19] Koutsonikola, V., ja Vakali, A. LDAP: framework, practices, and trends. *Internet Computing, IEEE 8*, 5 (syyskuu-lokakuu 2004), 66–72.

[20] Martins, V., Pacitti, E., ja Valduriez, P. Survey of data replication in P2P systems. Rapport de recherche RR-6083, INRIA, 2006.

[21] Myers, J. G. IMSP – Internet Message Support Protocol. Internet Draft, maaliskuu 1994.

[22] Newman, C., ja Myers, J. G. ACAP – Application Configuration Access Protocol. RFC 2244 (Proposed Standard), marraskuu 1997.

[23] OpenLDAP. OpenLDAP Software 2.4 Administrator's Guide. `http://www.openldap.org/doc/admin24/OpenLDAP-Admin-Guide.pdf`. Viitattu 28.2.2013.

[24] OpenLDAP. slapd-ndb. Linux man page, OpenLDAP 2.4.32.

[25] OpenLDAP. slapo-rwm. Linux man page, OpenLDAP 2.4.32.

[26] Oracle. Oracle Directory Server Enterprise Edition 11g Release 1 (11.1.1.3+) Certification Matrix. `http://www.oracle.com/technetwork/middleware/downloads/odsee-11gr1certmatrix-161592.xls`. Viitattu 5.3.2014.

[27] Oracle. Oracle Directory Server Enterprise Edition Data Sheet. `http://www.oracle.com/technetwork/middleware/id-mgmt/overview/ds-oracle-dsee-final-128269.pdf`. Viitattu 26.2.2013.

[28] Oracle. Oracle Fusion Middleware 11g Release 1 (11.1.1.x) Certification Matrix. `http://www.oracle.com/technetwork/middleware/downloads/fmw-11gr1certmatrix.xls`. Viitattu 5.3.2014.

[29] Oracle. Oracle Internet Directory 11g Data Sheet. `http://www.oracle.com/technetwork/middleware/id-mgmt/overview/oid-ds-11g-130905.pdf`. Viitattu 26.2.2013.

[30] Oracle. Oracle Unified Directory (OUD) 11gR1 (11.1.1.x) Certification Matrix. `http://www.oracle.com/technetwork/middleware/downloads/oud-11gr1certmatrix-404871.xls`. Viitattu 5.3.2014.

[31] Oracle. Oracle Unified Directory White Paper. `http://www.oracle.com/technetwork/middleware/id-mgmt/overview/whitepaper-oud-434007.pdf`, 2012. Viitattu 26.2.2013.

[32] Pedersen, T., ja Jensen, C. S. Multidimensional database technology. *Computer 34*, 12 (joulukuu 2001), 40–46.

[33] Perreault, S. vCard Format Specification. RFC 6350 (Proposed Standard), elokuu 2011.

[34] Potlog, A.-D., Xhafa, F., Pop, F., ja Cristea, V. Evaluation of Optimistic Replication Techniques for Dynamic Files in P2P Systems. In *Proceedings of the 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (Washington, DC, USA, 2011), 3PGCIC '11, IEEE Computer Society, s. 259–265.

[35] Rappaport, T. *Wireless Communications: Principles and Practice*, second ed. Prentice Hall PTR, 2002.

[36] Saito, Y., ja Shapiro, M. Optimistic Replication. *ACM Comput. Surv. 37*, 1 (maaliskuu 2005), 42–81.

[37] Sermersheim, J. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (Proposed Standard), kesäkuu 2006.

[38] Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., ja Hauser, C. H. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. *SIGOPS Oper. Syst. Rev. 29*, 5 (joulukuu 1995), 172–182.

[39] The Organization for the Advancement of Structured Information Standards (OASIS). Directory Services Markup Language v2.0. OASIS Standard, joulukuu 2001.

[40] Ullman, J. D. *Principles of Database and Knowledge-Base Systems*, vol. 1 of *Principles of Computer Science Series*. Computer Science Press, 1989.

[41] Wood, D. *Programming Internet Email*, first ed. O'Reilly, 1999.

[42] Zeilenga, K. The Lightweight Directory Access Protocol (LDAP) Content Synchronization Operation. RFC 4533 (Experimental), kesäkuu 2006.

Liite A

Simulaatioiden tulokset

Taulukko A.1: Lähetettyjen viestien määrät eri noodeilla täysin kytketyssä verkossa.

Noodi	Ajo 1	Ajo 2	Ajo 3	Ajo 4	Ajo 5	Ajo 6	Ajo 7	Ajo 8	Ajo 9	Ajo 10	Keskiarvo	Keskihajonta
1	9	34	9	10	17	10	10	10	8	9	12,6	7,9
2	8	18	9	9	18	7	7	9	7	9	10,1	4,3
3	7	26	15	8	33	5	6	8	11	7	12,6	9,5
4	13	38	17	9	17	10	10	6	14	15	14,9	8,9
5	35	26	16	40	7	28	30	37	4	35	25,8	12,7
6	4	11	42	2	23	3	3	2	22	4	11,6	13,4
Kaikki	76	153	108	78	115	63	66	72	66	79	87,6	28,9

Taulukko A.2: Vastaanotettujen viestien määrät eri noodeilla täysin kytketyssä verkossa.

Noodi	Ajo 1	Ajo 2	Ajo 3	Ajo 4	Ajo 5	Ajo 6	Ajo 7	Ajo 8	Ajo 9	Ajo 10	Keskiarvo	Keskihajonta
1	8	34	8	10	16	10	10	10	6	8	12,0	8,2
2	9	14	11	11	22	7	7	9	7	9	10,6	4,6
3	10	23	22	12	26	6	8	11	12	10	14,0	7,0
4	17	45	17	14	16	13	13	11	15	17	17,8	9,8
5	22	21	19	25	14	19	20	25	7	25	19,7	5,6
6	10	16	31	6	21	8	8	6	19	10	13,5	8,1
Kaikki	76	153	108	78	115	63	66	72	66	79	87,6	28,9

Taulukko A.3: Lähetettyjen viestien määrät eri noodeilla tähtiverkossa.

Noodi	**Ajo 1**	**Ajo 2**	**Ajo 3**	**Ajo 4**	**Ajo 5**	**Ajo 6**	**Ajo 7**	**Ajo 8**	**Ajo 9**	**Ajo 10**	**Keskiarvo**	**Keskihajonta**
1	32	32	32	30	32	50	38	32	30	30	33,8	6,1
2	5	3	5	2	5	6	3	2	2	2	3,5	1,6
3	3	6	6	5	6	6	4	6	4	4	5,0	1,2
4	4	4	4	4	4	8	6	3	5	3	4,5	1,5
5	6	4	2	3	2	8	8	4	6	6	4,9	2,2
6	3	5	4	6	4	7	6	6	3	5	4,9	1,4
Kaikki	53	54	53	50	53	85	65	53	50	50	56,6	10,9

Taulukko A.4: Vastaanotettujen viestien määrät eri noodeilla tähtiverkossa.

Noodi	**Ajo 1**	**Ajo 2**	**Ajo 3**	**Ajo 4**	**Ajo 5**	**Ajo 6**	**Ajo 7**	**Ajo 8**	**Ajo 9**	**Ajo 10**	**Keskiarvo**	**Keskihajonta**
1	21	22	21	20	21	35	27	21	20	20	22,8	4,8
2	8	4	8	2	8	10	4	2	2	2	5,0	3,2
3	4	10	10	8	10	10	4	10	6	6	7,8	2,6
4	6	4	6	6	6	10	10	4	8	4	6,4	2,3
5	10	6	2	4	2	10	10	6	10	10	7,0	3,4
6	4	8	6	10	6	10	10	10	4	8	7,6	2,5
Kaikki	53	54	53	50	53	85	65	53	50	50	56,6	10,9

Taulukko A.5: Lähetettyjen viestien määrät eri noodeilla rengasverkossa.

Noodi	**Ajo 1**	**Ajo 2**	**Ajo 3**	**Ajo 4**	**Ajo 5**	**Ajo 6**	**Ajo 7**	**Ajo 8**	**Ajo 9**	**Ajo 10**	**Keskiarvo**	**Keskihajonta**
1	9	19	7	5	11	15	8	5	6	7	9,2	4,6
2	8	19	5	3	12	15	9	4	5	6	8,6	5,2
3	13	16	17	9	14	17	12	11	14	15	13,8	2,6
4	15	16	18	7	13	15	16	13	17	17	14,7	3,2
5	16	18	18	6	11	13	14	17	18	18	14,9	4,0
6	14	16	18	9	11	14	13	14	15	18	14,2	2,8
Kaikki	75	104	83	39	72	89	72	64	75	81	75,4	16,9

Taulukko A.6: Vastaanotettujen viestien määrät eri noodeilla rengasverkossa.

Noodi	**Ajo 1**	**Ajo 2**	**Ajo 3**	**Ajo 4**	**Ajo 5**	**Ajo 6**	**Ajo 7**	**Ajo 8**	**Ajo 9**	**Ajo 10**	**Keskiarvo**	**Keskihajonta**
1	10	18	10	6	10	17	11	6	8	10	10,6	4,0
2	9	21	10	5	13	16	9	7	9	9	10,8	4,7
3	12	16	15	8	15	15	13	9	13	14	13,0	2,7
4	16	16	16	7	13	14	13	13	16	16	14,0	2,8
5	15	17	16	5	11	12	14	16	16	17	13,9	3,7
6	13	16	16	8	10	15	12	13	13	15	13,1	2,6
Kaikki	75	104	83	39	72	89	72	64	75	81	75,4	16,9

Taulukko A.7: Lähetettyjen viestien määrät eri noodeilla ketjuverkossa.

Noodi	**Ajo 1**	**Ajo 2**	**Ajo 3**	**Ajo 4**	**Ajo 5**	**Ajo 6**	**Ajo 7**	**Ajo 8**	**Ajo 9**	**Ajo 10**	**Keskiarvo**	**Keskihajonta**
1	4	4	4	5	4	5	3	6	2	2	3,9	1,3
2	9	9	13	16	9	16	6	18	10	7	11,3	4,2
3	8	9	15	18	9	19	7	18	18	12	13,3	4,8
4	10	12	15	18	9	19	13	19	19	10	14,4	4,1
5	14	10	11	16	9	16	14	18	18	9	13,5	3,5
6	5	3	3	6	4	5	5	7	6	4	4,8	1,3
Kaikki	50	47	61	79	44	80	48	86	73	44	61,2	16,7

Taulukko A.8: Vastaanotettujen viestien määrät eri noodeilla ketjuverkossa.

Noodi	**Ajo 1**	**Ajo 2**	**Ajo 3**	**Ajo 4**	**Ajo 5**	**Ajo 6**	**Ajo 7**	**Ajo 8**	**Ajo 9**	**Ajo 10**	**Keskiarvo**	**Keskihajonta**
1	6	6	6	8	6	8	4	10	2	2	5,8	2,6
2	8	8	12	14	8	15	5	16	11	7	10,4	3,7
3	6	8	15	17	8	17	7	17	17	11	12,3	4,7
4	10	11	14	17	8	17	11	16	17	10	13,1	3,5
5	12	10	10	15	8	15	13	17	16	8	12,4	3,3
6	8	4	4	8	6	8	8	10	10	6	7,2	2,1
Kaikki	50	47	61	79	44	80	48	86	73	44	61,2	16,7